INTERFÍCIE GRÀFICA PEL SIMULADOR CIENTÍFIC DE NANOESTRUCTURES BITLLES

Memòria del projecte de final de carrera corresponent als estudis d'Enginyeria Superior en Informàtica presentat per Arnau Padró Olivet i dirigit per Xavier Oriols Pladevall.

Bellaterra, juny de 2010

El firmant, Xavier Oriols Pladevall, professor del Departament d'Electrònica de la Universitat Autònoma de Barcelona

CERTIFICA:

Que la present memòria ha sigut realitzada sota la seva direcció per Arnau Padró Olivet

Bellaterra, juny de 2010

Firmat:

A tu, que m'has donat força per anar sempre endavant.

Agraïments

En primer lloc, agrair al meu tutor la seva tasca de guiatge i consell durant tot el desenvolupament del projecte. També a la meva família, que sempre m'ha fet costat i m'ha ajudat en tot el que ha calgut per tal d'assolir aquest objectiu, tan important per mi, de llicenciar-me en Enginyeria Informàtica. Finalment als companys de carrera, amb els quals he compartit preocupacions, pors i alegries, a més de fer-nos costat i ajudar-nos en els moments difícils.

Índex

1 Introducció **1**

1.1 Entorn del projecte 1
1.2 Motivacions 2
1.3 Descripció general 2
1.4 Objectius del projecte 3
1.5 Estructura de la memòria 4

2 Estudi de la viabilitat del projecte **7**

2.1 Estat de l'art 7
2.1.1 La indústria nanoelectrònica i la necessitat de nous simuladors de dispositius 8
2.1.2 Interfícies Gràfiques 9
2.2 Viabilitat tècnica 15
2.3 Viabilitat econòmica i legal 17
2.4 Viabilitat temporal 20

3 Anàlisi dels requeriments funcionals **23**

3.1 Requeriments funcionals 23
3.1.1 Identificació dels usuaris i subsistemes 23
3.1.2 Casos d'ús 24
3.1.3 Enumeració dels requeriments 26
3.2 Anàlisi funcional 28
3.2.1 Anàlisi funcional de la part de representació 3D 28
3.2.2 Anàlisi funcional de la comunicació nucli - interfície gràfica 31

3.2.3 Anàlisi funcional de la part de configuració de la simulació 32
3.2.4 Anàlisi funcional de la part de guardar/carregar dissenys . 33

4 Disseny i implementació software del projecte 35
4.1 Diagrama de classes UML 35
4.1.1 Classe Volum 36
4.1.2 Classe Material 39
4.1.3 Classe Property 40
4.1.4 Classe MainWindow 40
4.1.5 Classe GLWidget 42
4.1.6 Classes dels diàlegs de configuració 44
4.2 Disseny de la interfície 46
4.3 Estructures de dades per la comunicació nucli-interfície 49
4.4 Fitxers d'entrada i sortida 52

5 Tests 55
5.1 Test de multiplataforma 55
5.2 Test de la comunicació nucli-interfície 56
5.3 Test de realització d'un disseny real 58
5.4 Test de desat i càrrega d'un disseny real 59

6 Conclusions 61
6.1 Assoliment dels objectius 61
6.2 Futur i proposta d'ampliacions 64

Bibliografia 65

Índex de figures

2.1 Diagrama de Gantt de la planificació del projecte 20

3.1 Diagrama de Casos d'ús 24
3.2 Detallat del cas d'ús "creació d'un nou disseny" 25
3.3 Detallat del cas d'ús "edició d'un disseny existent" 26
3.4 Eixos de coordenades en OpenGL 29
3.5 Enllaços entre àtoms 31

4.1 Diagrama UML 37
4.2 Disseny de la finestra principal 47
4.3 Disseny del diàleg de configuració dels materials 48
4.4 Disseny del diàleg de configuració de les propietats generals del disseny 48
4.5 Disseny del diàleg de configuració de l'escala del disseny 49
4.6 Disseny del diàleg d'afegir/editar volums 50

5.1 La nostra aplicació funcionant sobre un sistema Ubuntu 56
5.2 La nostra aplicació funcionant sobre un sistema WindowsXP . . . 57
5.3 Test: Realització del disseny d'un Mosfet de dobla porta 58

6.1 Comparativa entre la planificació inicial i la real 63
6.2 Llistat de les tasques de la planificació 63

Índex de taules

2.1 Comparativa de Toolkits 16
2.2 Comparativa de les Llicències del toolkit Qt[27] 19
2.3 Tasques del projecte 21

3.1 Càlcul dels 8 vèrtex d'un paral·lelepípede a partir del seu vèrtex origen i la mida de les 3 dimensions 30

4.1 Estructures utilitzades per la comunicació nucli-interfície en C++ i FORTRAN 51
4.2 Estructura del fitxer de materials 54
4.3 Estructura del fitxer de propietats generals i dels volums 54
4.4 Estructura del fitxer del disseny 54

Capítol 1

Introducció

1.1 Entorn del projecte

Aquest projecte final de carrera de la titulació d'Enginyeria Informàtica s'engloba dins de les tasques de recerca que es desenvolupen dins del Departament d'Enginyeria Electrònica de la Universitat Autònoma de Barcelona. En particular forma, part del projecte d'investigació TEC2009-06986 subvencionat pel Ministeri de Ciència i Innovació.
Aquest projecte d'investigació té una durada de 3 anys i el seu objectiu és el desenvolupament d'un simulador de transport electrònic en nanoestructures amb la maduresa suficient per ser comercialitzat. Els fonaments teòrics del projecte d'investigació estan basats en l'article *"Quantum trajectory approach to time dependent transport in mesoscopic systems with electron-electron interactions"* [1] publicat a la prestigiosa revista Phys. Rev. Let. El nom oficial del projecte és BITLLES, que és l'acrònim de Bohemian Interacting Transport in Electronic Structures amb l'afegitó de la LL per donar-li sentit. Aquest simulador s'està desenvolupant utilitzant el llenguatge FORTRAN i està en un estat força avançat. Una de les parts més importants del projecte és l'entrada de dades pel disseny de les simulacions i la posterior verificació de l'estructura nanoelèctrica per comprovar que compleix els requeriments per ser simulada. Aquesta funcionalitat, encara no està implementada de manera eficient per tal de facilitar la tasca a l'usuari final.

És per això, que és necessari el desenvolupament d'una interfície gràfica pel simulador BITLLES que permeti a l'usuari dissenyar les simulacions de forma fàcil i intuïtiva. Aquesta és la finalitat d'aquest projecte final de carrera.

1.2 Motivacions

L'elecció d'aquest projecte d'entre altres opcions va comportar una reflexió d'uns dies. Finalment, em vaig decantar per aquest, ja que el disseny d'una interfície és una tasca que trobo interessant pel fet de ser el que l'usuari acaba percebent amb més claredat. Un bon disseny d'aquesta té un gran impacte en la productivitat i satisfacció de l'usuari. A més, el disseny d'interfícies, és un tema que s'ha tractat poc durant la carrera i vaig creure convenient aprofundir-hi. Per altra banda, el formar part d'un projecte de recerca amb projecció de futur va despertar el meu interès. Vaig pensar que si en un futur el meu treball era comercialitzat juntament amb la resta del projecte de simulador BITLLES podria ser valorat positivament en les meves aspiracions laborals.

1.3 Descripció general

El projecte que desenvoluparé serà doncs, una interfície gràfica que permeti la preparació de les simulacions de manera fàcil i intuïtiva per l'usuari. Aquesta preparació la podem dividir en 3 tasques diferents que cal realitzar:

1. Una primera tasca és la de dissenyar l'estructura que volem simular a través de la inserció de volums d'un determinat material. L'usuari haurà de poder visualitzar com va quedant l'estructura en tot moment per facilitar-li el control del què està fent.

2. Una segona tasca és la de configurar els diferents paràmetres generals que té la simulació. A més també s'haurà de permetre configurar els paràmetres associats a cada material.

3. Finalment com a darrera tasca la interfície a de permetre fer una comprovació de l'estructura de la simulació que hem preparat per tal de garantir que compleix totes les regles necessàries perquè la posterior simulació s'executi sense problemes.

1.4 Objectius del projecte

L'objectiu principal, per tant el més important, d'aquest projecte de final de carrera és el de realitzar una interfície gràfica que faciliti el màxim a l'usuari la preparació de les nanoestructures i paràmetres necessaris del simulador BITLLES. A partir d'aquest objectiu sorgeixen tot un seguit de objectius secundaris que tenen també una gran importància i els quals no podrem obviar:

- Un objectiu col·lateral al de facilitar a l'usuari el disseny de les seves simulacions és la millora de la productivitat d'aquest. S'espera que amb la nova interfície el temps necessari per preparar-les sigui substancialment menor a l'actual.

- Un altre dels objectius que es vol acomplir és el de no limitar el possible ús de la interfície gràfica a un sol sistema operatiu i arquitectura concrets. Volem tenir l'opció a que sigui compatible amb el màxim de plataformes possible, per tal d'arribar al màxim nombre possible d'usuaris

- També es vol aprofitar el màxim el treball ja existent fins ara, en altres paraules, no es vol duplicar funcionalitats entre el nucli de simulació intensiva i la interfície. És per aquest fet, que caldrà dissenyar un sistema de comunicació bidireccional entre el nucli i la interfície.

- Per altra banda, és molt important la possible adaptació de la interfície a futures noves funcionalitats del nucli de simulació de la manera més fàcil possible, per tal de minimitzar l'esforç necessari per aquesta tasca i permetre als desenvolupadors del simulador BITLLES centrar-se en el què és realment el seu objectiu.

1.5 Estructura de la memòria

El contingut d'aquesta memòria es divideix en 6 capítols més la bibliografia i els annexos. Els 4 capítols centrals són els que expliquen els diferents passos seguits durant el desenvolupament del projecte. Per altra banda el primer capítol serveix per situar el lector en el context i finalitats del projecte. L'últim capítol em permetrà fer un anàlisi de la feina realitzada i del futur del projecte.

En la introducció de la memòria s'explica el context del projecte per situar el lector, seguidament s'enumeren les raons de l'elecció d'aquest projecte per part de l'alumne, posteriorment ja s'entra a descriure en què consisteix el projecte realitzat i els seus objectius. Finalment es realitza aquesta explicació de l'estructura de la memòria per facilitar la lectura d'aquesta.

En el segon capítol es fa un anàlisi de la viabilitat del projecte. En primer lloc, es fa un estudi de l'estat de l'art encarat als simuladors de dispositius nanoelectrònics i el de les interfícies gràfiques. A partir d'aquesta informació passarem a analitzar la viabilitat tècnica fent una selecció de les eines que utilitzarem per realitzar el projecte. Seguidament farem un anàlisi econòmic i legal, fixant-nos en les condicions d'ús i llicència de les eines seleccionades. Finalment passarem a l'anàlisi temporal enumerant les tasques a realitzar i fent una proposta de planificació del projecte.

En el tercer capítol entrarem en el que és ja l'anàlisi dels requeriments funcionals que té el nostre projecte. Un cop identificats els perfils d'usuari i casos d'ús podrem fer una primera enumeració dels requeriments, per posteriorment, passar a analitzar amb més profunditat els més importants.

En el quart capítol s'explicarà la proposta de disseny i implementació del projecte centrant-nos en el diagrama de classes del conjunt del projecte, les estructures de dades necessàries, el disseny de la interfície gràfica i l'estructura dels fitxers d'entrada i sortida.

En el cinquè capítol es parlarà dels diferents tests que s'han anat fent durant el desenvolupament del projecte i s'analitzaran els resultats obtinguts.

Finalment, en les conclusions farem un anàlisi de l'assoliment dels objectius que ens havíem marcat inicialment i plantejarem quines possibilitats de futur i millores pot tenir el projecte.

Capítol 2

Estudi de la viabilitat del projecte

En aquest capítol farem un estudi de la viabilitat de realitzar el projecte. Per començar farem un anàlisi de l'estat de l'art actual en relació a les tecnologies que el projecte avarca o requereix. Un cop situats en el present tecnològic, passarem a analitzar la viabilitat del projecte des de 3 punts de vista fonamentals. En primer lloc, ja coneixent de quines alternatives disposem, farem una selecció de les eines de desenvolupament de les que farem ús, tenint en compte que compleixen amb tots els requeriments tècnics. Seguidament, entrarem en l'àmbit legal i econòmic analitzant la llicència de les eines de les quals fem ús i els requisits econòmics per tal de concloure si s'adeqüen als objectius i possibilitats del projecte. Finalment, analitzarem la viabilitat temporal fent una planificació acurada de les diferents tasques que caldrà realitzar per completar el projecte durant el temps del que disposem.

2.1 Estat de l'art

En aquesta secció analitzarem l'estat de l'art en relació al projecte des dels dos vessants que el defineixen. En primer lloc des de la vessant de la indústria nanoelectrònica i els simuladors que s'estan desenvolupant per les necessitats d'aquesta. L'altra vessant és la de l'estat de l'art de les interfícies gràfiques actuals per fer-nos a la idea de quines eines i tecnologia disposem per desenvolupar la nostra

solució.

2.1.1 La indústria nanoelectrònica i la necessitat de nous simuladors de dispositius

Al llarg de més de quatre dècades, la indústria microelectrònica s'ha caracteritzat per una evolució exponencial de les prestacions dels seus productes [2] [3]. D'una banda, s'ha augmentat el nivell d'integració ("Moore's law"), la velocitat de commutació i la funcionalitat dels circuits integrats, i de l'altra, s'ha disminuït el consum d'energia i el cost per funció realitzada. La majoria d'aquestes evolucions han sorgit com a conseqüència directa de l'habilitat de la indústria electrònica per reduir cada vegada més les dimensions del MOSFET convencional. No obstant això, actualment, hi ha un ampli consens en què l'escalat geomètric del CMOS no és suficient per seguir garantint les prestacions esperades dels futurs dispositius electrònics [2] [3]. En aquest sentit, la comunitat científica ha identificat el domini "More Moore"que estudia com evolucionar els dispositius MOSFET tradicionals mitjançant un compromís entre l'escalat tradicional i la introducció de noves solucions tecnològiques (dielèctrics d'alta permitivitat, transistors amb diverses portes, silici estressat, portes metàl.liques, etc) que es coneix com a "equivalent scaling"[2]. Hi ha un ampli consens en què aquestes propostes són actualment les millors estratègies per a la indústria electrònica en el període 2007-2022 que prediu l'actual ITRS [2]. No obstant això, la comunitat científica es planteja alternatives completament diferents al MOSFET ja que l'escalat exigit per la llei de Moore a llarg termini (transistors de 4 nm de longitud física de canal per 2022 [2]) serà inaccessible tecnològicament i econòmica. En aquest sentit, el domini "Beyond CMOS"estudia dispositius electrònics emergents amb principis bàsics diferents als del MOSFET que el puguin superar en, almenys, alguna de les seves prestacions. Per exemple, s'estudien dispositius on la magnitud rellevant no sigui la càrrega electrònica, sinó l'espín electrònic (espintrònica), dispositius basats en transport en túnel com el RTD ("ressonant tunneling Diodes"), dispositius d'un sol electró ("single-electron devices"), etc. També es plantegen suports físics

diferents del silici bulk semiconductor, com els nanofils de silici o els Çarbon-based Nanoelectronics"(per exemple grafè i nanotubs de carboni). Actualment no és evident quina d'aquestes propostes podrà substituir al MOSFET. En qualsevol cas, es creu que algunes d'aquests dispositius emergents podran conviure amb estructures CMOS nanomètriques en un futur pròxim utilitzant tecnologies no necessàriament electròniques (MEMS, sensors, etc) juntament amb arquitectures noves (bio-inspirades, computació quàntica, etc) o nous connexionats (3D, "Silicon-on-Package") en el que s'ha anomenat el domini "More than Moore"[2] [3].

Aquesta evolució des de la microelectrònica a la nanoelectrònica implica la creació de nous simuladors de dispositius nanoelectrònics desenvolupats per la comunitat científica. Destaquem el simulador Damocles[4] desenvolupat per D. Fischetti que proporciona prediccions per l'empresa IBM, Silvaco [5] i Synopsys [6] que són programes comercials de Monte Carlo. El programari NEMO [7] [8], desenvolupat per S. Datta, R. Lake and G. Klimeck per estudiar "high speed electronics" amb RTDs per a Texas Instruments. A nivell europeu destaquem NEXTNANO [9] desenvolupat pel grup del P. Vogl, del Walter Schottky Institute, i el TiberCad [10] del Group of Aldo Di Carlo, del Department of Electronics Engineering of the "Università di Rome Tor Vergata".

2.1.2 Interfícies Gràfiques

Les interfícies gràfiques van néixer a principis dels anys 70 com un pas endavant front de les interfícies de línia de comandes que s'havien utilitzat fins aquell moment. Al 1973, l'empresa Xerox PARC va desenvolupar Alto, el primer ordinador personal amb interfície d'usuari gràfica i el paradigma d'escriptori[11]. A partir d'aquí i a través sobretot de les empreses Apple i Microsoft i els seus sistemes operatius, pensats per apropar els ordinadors a persones amb pocs coneixements informàtics, les interfícies d'usuari es van anar expandint i evolucionant paral·lelament al creixement constant dels ordinadors personals en les llars i la majoria d'àmbits on es requeria tractament de dades. Seguidament farem un anàlisi de les opcions més solides i utilitzades que existeixen avui en dia per desenvolu-

par una interfície gràfica. A part, també analitzarem les dos principals alternatives en quan a desenvolupament d'interfícies 3D, ja que dins de la interfície gràfica hi haurem d'incloure una interfície 3D que ens permeti mostrar les estructures nanoelèctriques de la simulació. Aquestes alternatives són el DirectX i l'OpenGL.

Cocoa

El framework Cocoa[12] és el principal framework de desenvolupament per aplicacions de Mac OS. Aquest ens permet crear aplicacions gràfiques natives a la guia d'interfícies d'Apple a més de disposar de diferents nuclis per l'apartat d'àudio, vídeo, integració amb OpenGL, animacions, transparència en Internet i xarxes i suport per diferents llenguatges d'scripting a través de bindings. Utilitza un paradigma de disseny Model-View-Controller (MVC) que permet separar les dades de l'aplicació (Model) de la part gràfica (View) que ens la mostra i ens permet editar-les i controlar-ho tot plegat a través del Controller que programem.

GTK+

Les llibreries GTK+[13] són un toolkit[1] de disseny d'interfícies gràfiques multiplataforma programat en llenguatge C i que incorpora nombroses funcionalitats. Es publiquen sota la llicència GNU LGPL 2.1 la qual permet desenvolupar tan aplicacions lliures com propietàries (amb el codi font tancat), sense haver de pagar cap tipus de regalia. Com hem dit aquestes llibreries són multiplataforma i concretament poden utilitzar-se ens sistemes Windows, Mac OS, Linux i la majoria de sistemes basats en Unix. A més, s'utilitzen pel disseny de la interfície d'alguns mòbils avançats de Nokia i en el projecte One Laptop Per Child Project[14].
Té suport d'aspecte natiu en els sistemes operatius suportats, de temes, per a la internacionalització i localització de la interfície, del sistema de codificació de text UTF8, de text bidireccional (LTR/RTL), d'accessibilitat, d'autogeneració de documentació i molts altres.
A part de dels usos anteriorment esmentats també s'utilitza pel desenvolupament

[1] "Un toolkit és un conjunt de llibreries amb objectius diversos, que faciliten la tasca del desenvolupador"

de l'entorn d'escriptori lliure Gnome, que és utilitzat per defecte en el sistema operatiu Ubuntu; a més, s'utilitza en aplicacions conegudes com ara les versions Linux de Firefox i Nero, l'editor d'imatges Gimp o el client de missatgeria instantània multiprotocol Pidgin.

Interfícies Web

Amb la creixent expansió d'Internet i la millora de la qualitat i ample de banda de les seves xarxes s'està veient cada cop més com una alternativa l'ús d'interfícies web per aplicacions que fins fa pocs anys es desenvolupaven amb interfícies gràfiques locals sota el paradigma d'escriptori. La gran avantatge d'aquest canvi és que l'usuari pot accedir a aquestes aplicacions web i les dades que hi ha darrera emmagatzemades des de qualsevol dispositiu i lloc del món amb accés a Internet. Aquest canvi de paradigma forma part del Web 2.0[24] i està basat en tot un seguit de tecnologies que han anat sorgint com poden ser l'AJAX, que permet desenvolupar aplicacions web que s'executin des del client mantenint una connexió asíncrona amb el servidor en segon pla per fer peticions de dades i així evitar refrescar la pàgina; el Ruby on Rails, que tracta de combinar simplicitat amb la possibilitat de desenvolupar aplicacions web amb menys codi que altres framework; el ASP.NET, que és un framework creat per Microsoft per desenvolupar aplicacions web utilitzant qualsevol llenguatge suportat per .NET; el Java Web Start, creat per Sun Microsystems per tal de poder executar aplicacions Java directament des d'Internet utilitzant un navegador; el PHP, que és un llenguatge d'scripting molt estès pensat per crear webs dinàmiques; el JavaScript, que és una implementació de l'estàndard ECMAScript pensat per poder accedir els objectes computacionals que formen les webs; i un llarg etcètera.
El principal impulsor de les aplicacions web que podem trobar actualment és Google, pel seu interès de portar el major nombre possible d'usuaris en el seu entorn habitual, que no és altre que Internet. Google té un ampli ventall d'aplicacions web que ens serveixen d'exemple començant pel seu famós client de correu GMail, el visualitzador de mapes Google Maps, el paquet d'ofimàtica Google Docs que ens permet tractar tot tipus de documents o el lector de notícies Google

Reader.
A part de les aplicacions web de Google n'hi ha moltes altres. Per exemple tenim el eyeOS que ens permet tenir accés a un escriptori virtual amb un conjunt d'aplicacions de sèrie que van des d'un processador de textos, un calendari, un gestor de fitxers, un navegador, etc. Un altre exemple podria ser Flicker, que és un servei web que et permet gestionar i compartir la teva biblioteca de fotografies a més de disposar d'un editor d'imatges integrat.

Microsoft Foundation Class Library

Les Microsoft Foundation Class Library (MFC)[19] són unes llibreries que inclouen amplies porcions de la Windows API en classes de C++. Existeixen classes per la majoria d'objectes Windows handle-managed, a més de finestres predefinides i controls comuns. Aquestes llibreries van ser introduïdes l'any 1992 juntament amb el compilador Microsoft's C/C++ 7.0. Algunes de les seves característiques són els macros per capturar missatges de Windows, excepcions, identificacions de tipus en temps d'execució, serialització i inicialització dinàmica de classes. Aquestes llibreries són les més utilitzades en el desenvolupament d'aplicacions per sistemes Windows. Els principals exemples els podem trobar en les mateixes aplicacions que desenvolupa Microsoft com el paquet d'ofimàtica Office, el client de missatgeria instantània Windows Live Messenger o el Windows Media Player.

Qt

Les Qt[18] són un toolkit de desenvolupament d'interfícies gràfiques multiplataforma, que actualment desenvolupa a Nokia després de comprar Trolltech, l'empresa creadora d'aquest toolkit. Les Qt s'utilitzen amb el llenguatge de programació C++, però a més incorpora un pre-processador anomenat Meta Object Compiler per enriquir el llenguatge. A més, té nombrosos binding que permeten utilitzar la llibreria en nombrosos llenguatges de programació, entre ells Java, Python i PHP. Aquest framework té un ampli ventall de funcionalitats, com per exemple, suport per gràfics 3D a través de OpenGL, suport de gràfics 2D, suport per aplicacions

multifil, un subsistema multimèdia anomenat Phonon que permet reproducció de la majoria de formats amb facilitat, suport per tractar XML, suport per Scripting a través del subsistema QtScript, suport de xarxa, integració amb l'enginy HTML Webkit, suport per bases de dades, etc. Com hem dit, les Qt són multiplataforma i estan disponibles pels sistemes Windows, Mac Os, Linux i la majoria de sistemes basats en Unix. A més s'utilitzen en alguns dispositius mòbils com per exemple el Nokia N900 o bé el Motorola A760, amb una possible expansió futura per ser el toolkit per defecte del projecte de sistema operatiu per a dispositius mòbils MeeGo[20].
Històricament les Qt sempre han aplicat una política de doble llicència que permetia al usuari escollir segons el què li interesses. En primer lloc, tenia l'opció d'utilitzar les Qt amb la llicència GPL de manera gratuïta amb la limitació que només podia desenvolupar aplicacions de codi font obert. En el cas de voler fer una aplicació comercial de codi font tancat havia de comprar les Qt amb llicència Qt Commercial Developer License, que és una adaptació de l'anterior amb l'excepció de poder enllaçar amb programes de codi font tancat. Des de la versió 4.5, però, s'ha afegit una tercera possibilitat, que és la d'utilitzar les Qt amb la llicència LGPL, que si que permet enllaçar aquestes amb programes de codi font tancat de manera gratuïta.
Alguns exemples d'aplicacions desenvolupades amb aquesta tecnologia són el visor de mapes Google Earth, el client de VOIP Skype, el client de màquines virtuals Virtual Box, l'escriptori lliure KDE, el navegador Opera o el reproductor multimèdia VLC Media Player.

Swing (Java)

El Swing[17] és un toolkit de disseny d'interfícies per Java. És part de la Java Foundation Classes (JFC). Igual que el Cocoa segueix un model MVC i està pensat per ser usat amb un sol fil d'execució. Al utilitzar-se sobre Java adquireix els avantatges d'aquest. Els principal són el suport multiplataforma i la gran preparació per ser utilitzat a través d'Internet gràcies a la seva Java Virtual Machine (JVM). Les principals característiques que té són la seva extensibilitat per-

meten l'adaptació a implementacions personalitzades d'interfícies específiques, a més permet la personalització de l'aspecte visual modificant diferents elements com ara les cantonades, el fons dels widgets, el color, opacitats, etc. Un dels principals problemes que té Swing és la dificultat de debbugar les aplicacions a causa del buffer intermig entre l'aplicació i la pantalla. Aplicacions fetes en Swing són per exemple els entorns de desenvolupament NetBeans i Eclipse, el client de torrent Azureus o l'editor de gràfics SVG Sketsa, a més d'un munt d'aplicacions web.

DirectX

Les DirectX[15] són unes APIs de Microsoft per tasques relacionades amb el multimèdia, sobretot per la programació de jocs sobre sistemes Microsoft. És present en els sistemes operatius Windows des de la seva versió Windows 95 i fins l'actualitat. Aquesta API inclou diferents subsistemes com per exemple el Direct3D, encarat a gràfics 3D; el DirectSound, encarat a so; el DirectDraw a gràfics 2D o el DirectPlay, encarat a comunicacions en xarxa que facilita la programació de jocs en línia. A part de ser utilitzada en el desenvolupament de jocs d'ordinador, també s'utilitza en el desenvolupament dels jocs per la consola Xbox i la nova Xbox 360, cosa que fa que bona part dels jocs actuals siguin fets amb aquesta tecnologia.

OpenGL

OpenGL[16] és una API lliure i estanderditzada multillenguatge i multiplataforma (fins hi tot existeix una versió per mòbils anomenada OpenGL ES) pensada pel disseny d'aplicacions que produeixen gràfics 3D. Inicialment fou desenvolupada per Silicon Graphics Inc. (SGI) al 1992. Ofereix al programador una API senzilla, estable i compacta. A més la seva escalabilitat ha permès que no s'hagi estancat el seu desenvolupament i també la creació d'extensions, una sèrie d'afegits sobre les funcionalitats bàsiques, per tal d'aprofitar les creixents evolucions tecnològiques. Podem ressenyar la inclusió del GLSL (un llenguatge de shaders propi) com a estàndard en la versió 2.0 d'OpenGL, publicada l'any 2004. És molt utilitzada en aplicacions CAD, de realitat virtual, de visualització científica i també en el

desenvolupament de jocs, sobretot en aquells que es vol que estiguin disponibles en diverses plataformes. En els sistemes amb plataforma Microsoft competeix amb DirectX. Tan l'iPhone, com els sistemes Android i Symbian OS utilitzen l'OpenGL ES com la seva principal llibreria gràfica[21].

2.2 Viabilitat tècnica

Per tal de realitzar el projecte, necessitarem un seguit d'eines que s'adaptin als requeriments que tenim. Seguidament farem una relació de les eines que necessitem juntament amb els requeriments que hauria de tenir cadascuna:

- Un toolkit pel desenvolupament de la interfície gràfica de l'aplicació, que sigui multiplataforma, per tal de complir amb l'objectiu de que la nostra aplicació sigui compatible amb el màxim de plataformes, sobretot pensant en sistemes Windows, Linux i Mac Os. A més, el toolkit hauria de ser el màxim de compatible amb l'ús d'algunes de les tecnologies per la producció de gràfics 3D.

- Haurem d'escollir entre l'ús de DirectX o bé OpenGL per a la realització de la representació 3D de les nanoestructures elèctriques.

- Un compilador multiplataforma del llenguatge que utilitzem per a la realització del projecte, tenint en compte que aquest llenguatge ha de ser compatible amb el toolkit que finalment escollim.

- Un compilador multiplataforma de FORTRAN per poder compilar el nucli de simulació.

Tenint en ment aquests requeriments es van analitzar les diferents opcions plantejades en la secció anterior. En primer lloc es van descartar les alternatives que no eren multiplataforma (Cocoa i Microsoft Foundation Class Library) ja que limitaven la disponibilitat de la nostra aplicació a un sol sistema operatiu (Mac OS i Windows respectivament). Per la mateixa raó vam descartar l'ús de DirectX per

Taula 2.1: Comparativa de Toolkits

	Cocoa	**Gtk+**	**Web**	**MFC**	**Qt**	**Swing**
Multiplataforma	No	**Sí**	**Sí**	No	**Sí**	**Sí**
Suport OpenGL	**Sí**	**Sí** Amb 1 extensió	**Sí** Poc Madur	**Sí**	**Sí**	**Sí** Amb 1 extensió
Llenguatge(s) Natius	Objective-C C C++	C	JavaScript Python PHP Ruby	C++	C++	Java

a la generació dels gràfics 3D i per tant ens vam decantar per l'única alternativa existent amb qualitat suficient i que sí que era multiplataforma, l'OpenGL. Seguidament, vam descartar realitzar una interfície web ja que la integració d'aquestes amb l'OpenGL tot i existir[23], està en una fase molt prematura, i a més pel model d'aplicació (ens sentíem còmodes amb el model tradicional) que nosaltres volíem tampoc ens aportava grans avantatges. Seguint aquest mateix criteri vam acabar decidint-nos per les llibreries Qt, perquè de les 3 opcions restants era la que tenia una millor integració amb OpenGL. Les Qt disposen d'un Widget per aquest propòsit. En el cas de Swing, tot i que el llenguatge de programació Java dona una experiència superior al programador en comparació amb els llenguatges C i C++, el fet de que la implementació OpenGL per Java no formes part de les Java Foundation Classes va ser un punt negatiu. Pel que fa a les Gtk+, tot i tenir una extensió que dona suport a OpenGL[22] aquesta no forma part del projecte principal. A més, les Gtk+, en comparació amb les Qt tenen unes característiques tècniques inferiors, suporten menys tecnologies. Un últim element que crec que és important, és el fet que ja tenia certa experiència amb el toolkit Qt al haver desenvolupat algunes petites aplicacions en el meu temps de lleure, sent un altre punt a favor d'aquesta elecció.

És per tot això que la nostra selecció es va decantar per utilitzar el toolkit Qt, ja que complia perfectament els nostres requeriments d'adaptar-se plenament a OpenGL i de ser multiplataforma. A més, el toolkit porta integrat un entorn de desenvolupament força potent que permet dissenyar la interfície, programar el seu funcionament i debbugar l'aplicació.

Un cop escollits el toolkit i l'API per la generació de gràfics 3D ens mancava escollir el(s) compilador(s) pel llenguatge C++, que és el llenguatge que utilitzen les llibreries Qt, i pel llenguatge FORTRAN. Aquesta decisió, cal dir-ho, va ser força més senzilla que l'anterior. Vam escollir el compilador GCC, sense examinar massa altres alternatives, pel fer d'adaptar-se al 100% als nostres requeriments:

1. El compilador GCC és multiplataforma, suportant Windows, Mac OS, Linux i la majoria de sistemes basats en Unix.

2. El compilador GCC suporta tan la compilació de llenguatge C++ com FORTRAN a través dels seus plugins g++ i gfortran.

3. El compilador GCC s'integra perfectament en l'entorn de desenvolupament que ens proporcionen les llibreries Qt, de fer és el compilador que utilitza per defecte.

4. Finalment, cal destacar, que aquest compilador té la capacitat d'enllaçar binaris provinents de codi font fet en C++ i en FORTRAN, cosa que ens serà de gran utilitat per no duplicar codi font. Això s'aconsegueix gràcies a un codi intermig que genera el compilador des de qualsevol dels llenguatges que suporta a través del seu sistema de plugins.

Podem concloure, doncs, que el nostre projecte és viable tècnicament ja que hem trobat eines que complien els requeriments tècnics que ens havíem marcat per tal de poder dur-lo a terme. A partir d'aquí caldrà analitzar la resta d'aspectes per acabar de concloure la viabilitat del projecte.

2.3 Viabilitat econòmica i legal

Un cop feta la selecció de les eines que utilitzarem a partir de criteris estrictament tècnics cal analitzar també sí s'adeqüen a la nostra capacitat econòmica i si tenim algun tipus de limitació legal que afecti les nostres aspiracions.

En l'apartat econòmic, tan el compilador gcc (i els seus plugins g++ i gfortran) com les implementacions de la API de OpenGL que pugui tenir cada sistema operatiu en principi no suposen cap cost econòmic pel nostre projecte. Per altra banda, en el cas de les llibreries Qt, el cost econòmic és diferent segons el tipus de llicència que volguem que tingui. Actualment té una versió gratuïta que es publica sota la llicència LGPL[2] i per altra banda té una versió de pagament, la qual et garanteix suport tècnic de la companyia, amb llicència Qt Commercial Developer License.

En l'apartat legal, el més important és fixar-nos en quin tipus de llicència tenen les diferents eines que utilitzarem. El cas més rellevant és el de les llibreries Qt. Tal i com hem comentat anteriorment, les Qt tenen una política de triple llicència tal com es pot veure a la taula 2.2. Les tres llicències que ens ofereix són la GPL[3], la LGPL (des de la versió 4.5) i la Qt Commercial Developer License.

La llicència LGPL[25] fou creada per GNU juntament amb la llicència GPL[26] per tal de promoure la seva filosofia del programari lliure. Aquesta filosofia propugna que el codi font dels programes ha d'estar disponible per a tothom i no només això, sinó que a més s'ha de permetre que qualsevol pugui modificar-lo i distribuir les seves millores, cosa que creuen, seria molt beneficiós per la llibertat de l'usuari. Dins d'aquest context varen crear la llicència GPL que garantia aquests drets, i a més, conté el que anomenen el "caràcter víric". Això suposa que un projecte que enllaci amb alguna llibreria que estigui sota aquesta llicència es veurà obligat a tenir aquesta mateixa llicència o una de compatible, o sigui que respecti les condicions que imposa d'ús, modificació i distribució del codi. Per altra banda, la llicència LGPL té les mateixes condicions que la GPL sense imposar el caràcter víric als programes que enllacin amb el què està sota aquesta llicència. Aquest petit detall, fa que ens decantem per la versió LGPL de les llibreries Qt, que a més, és gratuïta, sense patir pel fet de publicar el nostre programa amb una llicència que no compleixi amb les condicions que ens imposa la GPL. És un fet molt important ja que ens limitaria la possible comercialització del simulador a

[2]"Acrònim de Library General Public License"
[3]"Acrònim de General Public License"

Taula 2.2: Comparativa de les Llicències del toolkit Qt[27]

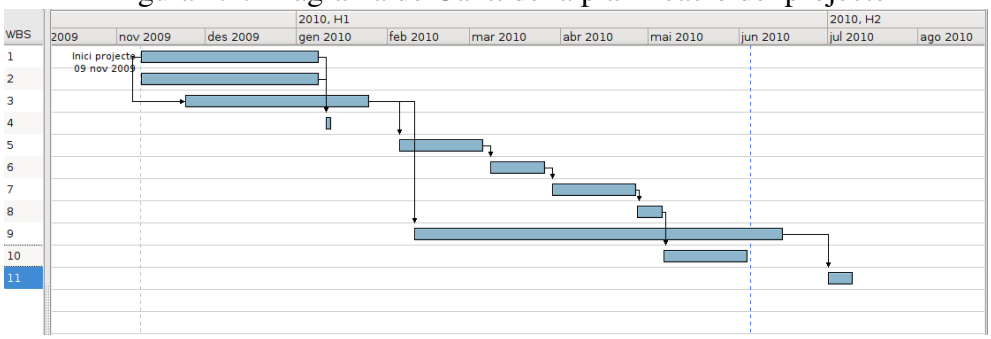
	Comercial	**LGPL**	**GPL**
Cost de llicència	Sí	No	No
S'han de proporcionar els canvis al codi font de les Qt?	No	Sí	Sí
Es poden crear aplicacions de codi font tancat	Sí	Sí	No
Es poden crear aplicacions de codi font tancat	Sí, No cal proporcionar codi font	Sí, En concordança amb els termes de la LGPL 2.1	No, les aplicacions creades estan subjectes a la publicació del seu codi font
Actualitzacions proporcionades	Sí, es notifiquen immediatament	Sí	Sí
Suport tècnic disponible	Sí	No, però es pot adquirir separadament	No, però es pot adquirir separadament
Taxa per runtime	Sí, per alguns sistemes incrustats	No	No

no ser que permetéssim la distribució del seu codi font. Aquesta és una decisió que es prendrà en un futur i no convenia que ara ens limitéssim les possibles opcions. L'únic efecte que té la LGPL és el fet que si voléssim modificar les pròpies llibreries Qt per adaptar-les a les nostres necessitats, n'hauríem de fer públiques aquestes millores. Tot i així, és complicat haver d'arribar a aquest extrem, i en tot cas aquest fet tampoc tindrà un efecte massa important en la possible comercialització de l'aplicació.

També tenim l'opció d'escollir la versió de les Qt sota llicència Qt Commercial Developer License, si veiem que ens interessa. Aquesta versió, a més de donar-nos suport tècnic, com hem comentat anteriorment, ens elimina l'obligació de publicar els possibles canvis que podéssim fer a les llibreries Qt.

Per tant, com podem veure, el nostre projecte és viable tan econòmica, com legalment utilitzant les eines que ja havíem escollit. L'únic interrogant important era quina versió de les llibreries Qt ens afavoria més, i en principi la millor opció és la que està publicada sota la llicència LGPL, ja que ens evita tenir costos i a més, no ens limita el possible futur comercial del projecte. Ara només caldrà fer un anàlisi de la viabilitat temporal del projecte planificant les tasques a realitzar durant el temps del que disposem.

Figura 2.1: Diagrama de Gantt de la planificació del projecte



2.4 Viabilitat temporal

Per garantir la viabilitat temporal del projecte el més important és tenir clares les tasques en les que es dividirà i quina quantitat de temps requeriran. Un cop fet això, caldrà programar-les al llarg del període en que es desenvoluparà el projecte, per tal de garantir la seva finalització. A la taula 2.3 podem veure la relació de les tasques que pensem caldrà realitzar per completar el projecte. Com podem veure estan numerades i a més del nom hi ha una petita descripció, la durada i les tasques de les quals depèn. Cal puntualitzar que la durada és aproximada i no representa el temps real de feina, sinó el marge de temps que crec necessari per completar la tasca. També s'ha tingut en compte que durant els primers 3 mesos la càrrega de treball serà inferior als altres 3 pel fet de disposar de menys temps. Per altra banda en l'apartat de predecessors AC significa "Acaba la tasca per començar" i CC significa "Començar la tasca per començar", referent-se a la tasca de la què depèn.

Un cop tenim les tasques enumerades i quantificades confeccionarem el diagrama de Gantt que ens permetrà tenir un esquema visual de la planificació del projecte a més de poder ajudar-nos en cas de possibles retrassos alhora de reestructurar les tasques. Per elaborar-lo s'han afegit alguns retrassos entre tasques dependents tenint en compte períodes de temps en que no serà possible destinar recursos temporals al desenvolupament del projecte. En la imatge 2.1 podem veure com queda la planificació final pel projecte.

Amb aquesta planificació finalitzem el capítol d'anàlisi de la viabilitat amb la

Taula 2.3: Tasques del projecte

Nº	**Tasca**	**Descripció**	**Durada**	**Depèn**
1	**Anàlisi funcional i de requeriments**	Per tal de conèixer els requeriments del projecte es faran un seguit de reunions amb el tutor, que alhora és el client, per anar-los concretant.	1 mes i mig	
2	**Investigació de l'estat de l'art**	Caldrà conèixer quines eines existeixen al mercat tan pel què fa a simuladors nanoelèctrics, com per les possibles eines que caldrà utilitzar.	1 mes i mig	
3	**Disseny de l'aplicació**	Un cop coneixem els requeriments i estat de l'art, ja podem realitzar el disseny de l'aplicació, el disseny de classes UML, disseny de l'aspecte gràfic, estructures de dades necessàries i format dels fitxers.	1 mes i mig	1 CC
4	**Redacció de l'informe previ**	Confecció de l'informe previ a partir del treball fet.	2 dies	1 i 2 AC
5	**Codificació de la interfície 3D**	A partir del disseny fet, caldrà codificar el funcionament de la interfície 3D i com es generen els gràfics de les nanoestructures elèctriques.	3 setmanes	3 AC
6	**Codificació de la interfície de configuració dels paràmetres de simulació**	A partir del disseny fet, caldrà codificar el funcionament de la interfície de configuració dels diferents paràmetres de la simulació.	2 setmanes	5 AC
7	**Codificació de la comunicació nucli - interfície**	A partir del disseny fet, caldrà codificar el funcionament de la comunicació nucli - interfície utilitzant les estructures de dades pensades.	3 setmanes	6 AC
8	**Codificació de desar/carregar dissenys.**	A partir del disseny fet i del format de fitxer pensat, caldrà codificar la funcionalitat de desar/carregar dissenys de simulacions a mig fer, o fetes.	1 setmana	7 AC
9	**Redacció de la memòria**	Un cop encarat el desenvolupament del projecte ja podrem començar a redactar la memòria que caldrà entregar al tribunal.	3 mesos	3 AC
10	**Depuració i correcció d'errors en l'aplicació**	Un cop acabada la codificació, caldrà testejar les diferents funcionalitats i comprovar-ne el correcte funcionament.	3 setmanes	8 AC
11	**Presentació del projecte**	Preparació de la presentació del projecte.	1 setmana	9 AC

clara conclusió que en termes tècnics, econòmics, legals i temporals el projecte pot tirar endavant.

Capítol 3

Anàlisi dels requeriments funcionals

3.1 Requeriments funcionals

En el capítol anterior hem definit ja un seguit de requeriments que afecten el projecte. Aquests requeriments eren bàsicament tècnics, relacionats principalment amb la selecció de les eines que faríem servir, ja que són els que ens defineixen les capacitats tècniques que aquestes han de tenir. En aquest capítol, doncs, ens centrarem en els requeriments funcionals del projecte. Ens referim a la manera com el nostre client ens ha demanat que ha de funcionar l'aplicació, segons les seves necessitats.

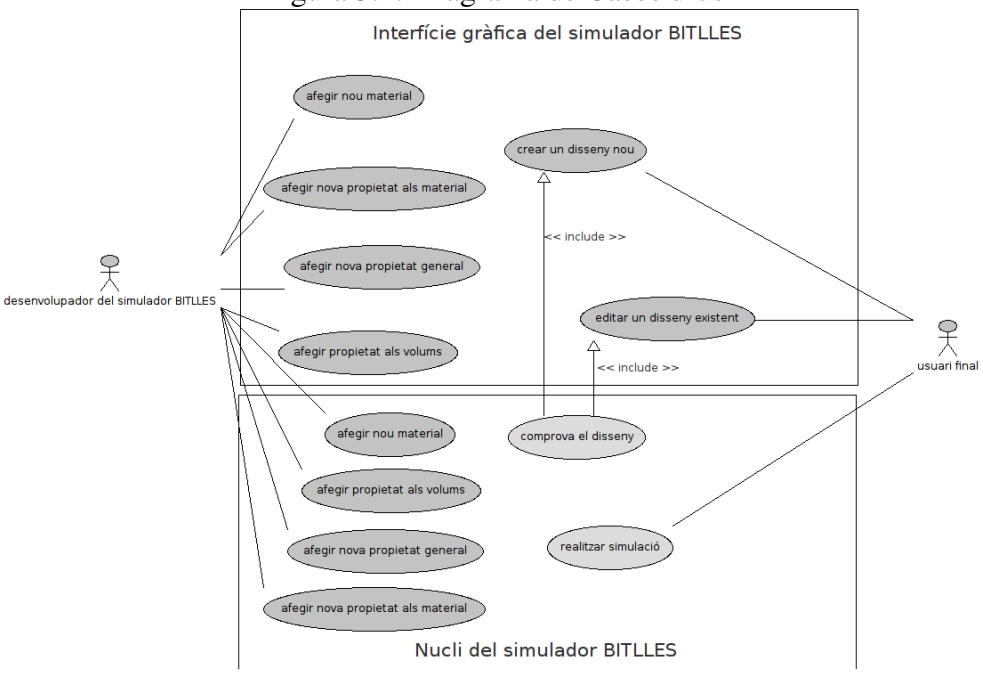
3.1.1 Identificació dels usuaris i subsistemes

El primer que cal fer és identificar quins usuaris tindrà la nostra aplicació. S'identifiquen dos tipus d'usuari molt ben definits:

1. **Desenvolupadors del simulador BITLLES:** Els desenvolupadors del simulador BITLLES són els primers implicats en la interfície que jo desenvolupo ja que en faran ús a través del seu nucli de simulació i per tant s'ha de tenir molt en compte els seus requeriments i necessitats.

2. **Usuari final del simulador:** Els possibles usuaris que utilitzin la nostra interfície gràfica juntament amb el nucli de simulació per a realitzar les

Figura 3.1: Diagrama de Casos d'ús



seves simulacions d'estructures nanoelèctriques que vulguin estudiar.

Per altra banda podem diferencia dos subsistemes diferents dins el simulador BITLLES:

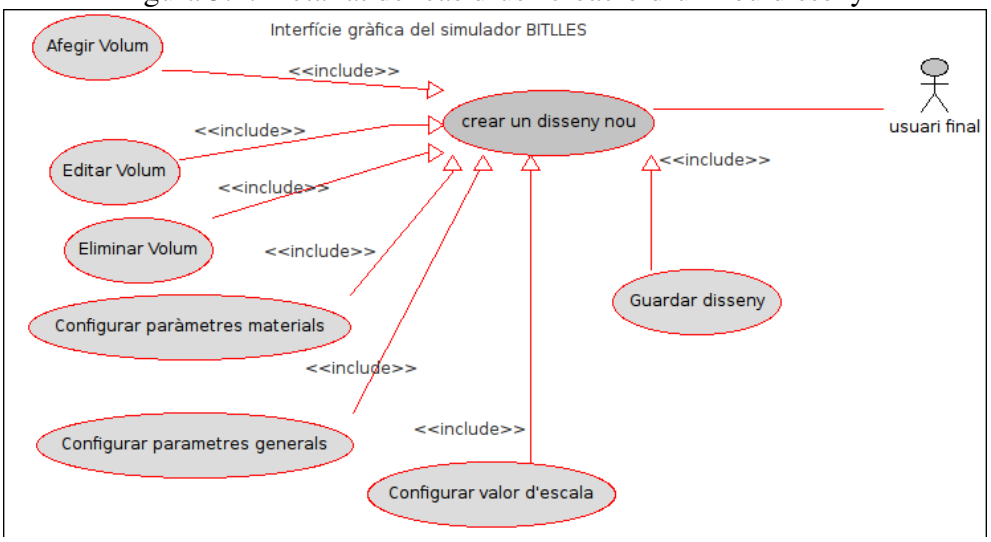
1. En primer lloc, tenim **la interfície gràfica**, que és l'element que nosaltres desenvoluparem.

2. Per altra banda, cal destacar l'existència del **nucli de simulació** pel fet que l'haurem de tenir en compte en el nostre disseny.

3.1.2 Casos d'ús

Un cop coneixem els usuaris i subsistemes del sistema passarem a especificar els casos d'ús d'aquest. Això ho farem a través d'un diagrama de casos d'ús el qual representa la forma com els diferents Actors (usuaris del sistema) operen amb aquest. La figura 3.1 representa el diagrama de casos d'ús de la nostra aplicació.

Figura 3.2: Detallat del cas d'ús "creació d'un nou disseny"



Casos d'ús de l'usuari final:

1. **Crear un nou disseny.** Té un seguit de subcasos inclosos que són els que permetran conformar el disseny. Aquests subcasos estan detallats a la figura 3.2.

2. **Editar un disseny ja existent.** Tal i com podem veure en la figura 3.3 aquest cas té un seguit de subcasos inclosos que són els que permetran editar el disseny. A més, té com a dependència "Carregar disseny existent", ja que és imprescindible per poder realitzar aquest cas d'ús.

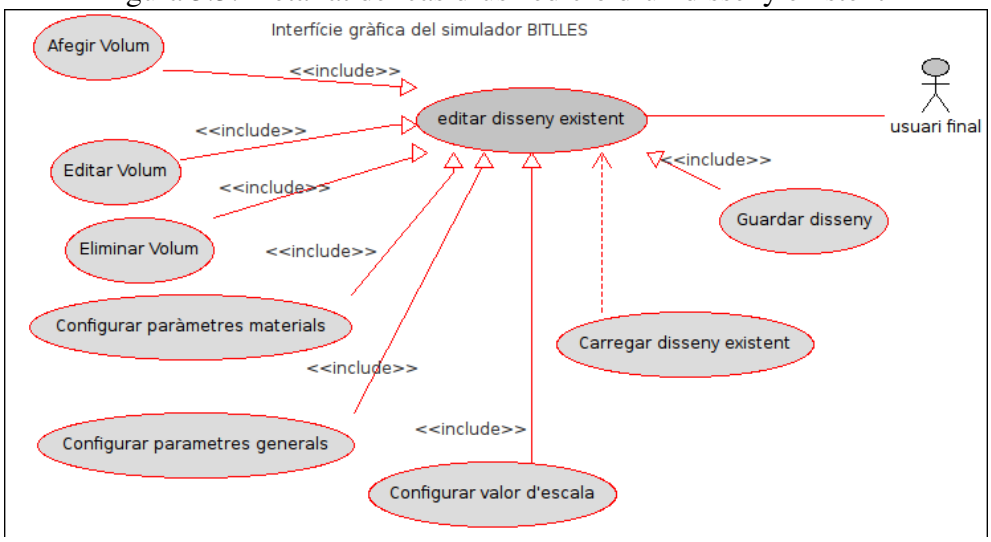
Casos d'ús del desenvolupador del projecte BITLLES:

1. **Afegir un nou material.** Permet afegir nous materials que estaran disponibles per l'usuari final per associar-los a volums.

2. **Afegir una nova propietat als materials.** Permet afegir noves propietats als materials.

3. **Afegir una nova propietat general de la simulació.** Permet afegir noves propietats generals de la simulació.

Figura 3.3: Detallat del cas d'ús "edició d'un disseny existent"



4. **Afegir una nova propietat als volums.** Permet afegir noves propietats als volums que l'usuari haurà de definir.

Com podem veure al diagrama, els casos d'ús que afecten el desenvolupador els ha de realitzar tan al nucli de simulació com a la interfície, ja que en cas contrari les noves funcionalitats afegides no estarien disponibles per l'usuari.

A més, cal destacar la dependència entre els casos d'ús de realitzar un nou disseny i editar-ne un d'existent amb un cas d'ús del nucli de simulació. Aquest fet caldrà ser tingut en compte a l'hora de dissenyar la interfície per tal d'evitar la duplicitat de codi.

3.1.3 Enumeració dels requeriments

Seguidament uns enumerem els requeriments que hem anat trobant en les reunions amb el client. Els dividirem entre els que afecten l'usuari final i els que afecten als desenvolupadors del projecte.

Requeriments que afecten l'usuari final:

- L'usuari ha de poder afegir, editar i eliminar volums al disseny de forma senzilla.
- Els volums representen a un material concret i tenen un seguit de característiques associades, com per exemple, la mida del volum, la posició dins el disseny, el nombre de malles i un nom que els identifiqui.
- L'usuari ha de poder veure en tot moment una representació gràfica 3D de l'estructura que està dissenyant.
- L'usuari ha de poder interactuar amb aquesta representació per veure-la des de qualsevol angle i distància possible.
- L'usuari ha de poder visualitzar la representació 3D de l'estructura tan en un mode de sòlids, com en un mode d'estructura atòmica.
- L'usuari ha de poder establir la unitat d'escala que s'aplica a les posicions i mides dels volums.
- L'usuari ha de poder configurar els paràmetres generals de la simulació, així com les propietats dels materials que pot associar als volums.
- L'usuari ha de poder comprovar si el seu disseny compleix amb les restriccions i regles que imposa el nucli de simulació.
- L'usuari ha de poder ocultar els volums que vulgui per tal de poder observar parts del disseny que quedin amagades.
- L'usuari ha de poder guardar i carregar els dissenys realitzats.

Requeriments que afecten el desenvolupador:

- Les propietats associades a un volum han de ser ampliables de manera fàcil per adaptar-les a futures necessitats del nucli intensiu de simulació.
- Els materials als quals es pot associar un volum han de ser ampliables de manera fàcil, així com les propietats que portin implícites aquests materials.

- La interfície gràfica i el nucli de simulació s'han de comunicar en dues ocasions. Quan la interfície requereixi comprovar el disseny, i quan es vulgui realitzar una simulació.
- Cal tenir en compte que els volums seran sempre paral·lelepípedes, ja que el nucli de simulació treballa només amb aquest tipus de formes.

3.2 Anàlisi funcional

En aquesta secció i a partir dels requeriments que tenim, aprofundirem en l'anàlisi funcional del projecte dividint-lo en les diferents parts que el conformen.

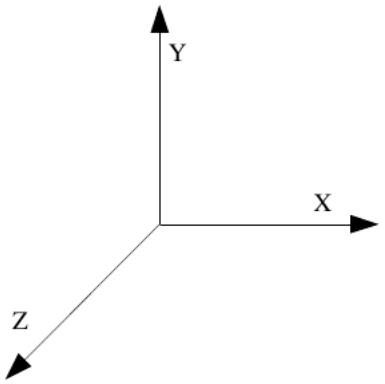
3.2.1 Anàlisi funcional de la part de representació 3D

Tal i com hem dit, aquesta part s'encarregarà de mostrar el conjunt de volums afegits per l'usuari que conformaran l'estructura nanoelèctrica. Per tant, s'haurà d'anar refrescant sempre que hi hagi un canvi en el conjunt de volums, tan si n'afegim, com si n'eliminem, com si n'editem algun.

En quan a la manera d'interactuar amb la representació per poder veure l'estructura des de diferents angles i distàncies, implementarem 3 tipus de d'accions diferents:

1. En primer lloc, l'usuari podrà fer zoom (augmentar i disminuir) a través de la rodeta del ratolí. D'aquesta manera l'usuari podrà tenir una visió més o menys global de l'estructura nanoelèctrica, segons li interessi, de forma ràpida i eficaç.
2. En segon lloc, l'usuari podrà fer rotar el disseny sobre el seu origen clicant sobre aquest amb el ratolí i movent-lo, mantenint el botó clicat.
3. Finalment, l'usuari podrà moure l'estructura a través dels cursos del teclat. Segons la direcció de la tecla que haguem pitjat, l'estructura es mourà en aquella direcció.

Figura 3.4: Eixos de coordenades en OpenGL



Un element important a tenir en compte és el sistema de coordenades amb el que treballarem. L'OpenGL té un sistema ortonormal (és a dir els seus eixos són perpendiculars entre ells) cartesià de 3 dimensions com el que es pot veure a la figura 3.4. Totes les transformacions 3D que ens permet fer OpenGL treballen amb aquest sistema de coordenades i ens permeten rotar, traslladar i escalar objectes. També cal tenir en compte que les diferents formes es dibuixen a partir dels seus vèrtex. OpenGL ens permet dibuixar punts, rectes, triangles i rectangles. A més a través de llibreries extra com les GLU podem dibuixar fàcilment esferes i altres formes 3D.

Com hem dit, hi haurà dos modes de visualització de l'estructura. El mode sòlid ens mostrarà els volums afegits al disseny en forma de paral·lelepípede sòlid. L'usuari ens haurà proporcionat un punt d'origen i unes mides (alçada, amplada i profunditat) sempre especificant les 3 components del sistema de coordenades. Amb aquestes dues dades haurem de calcular la resta de vèrtex (8 en total) per poder dibuixar les 6 cares que formen el paral·lelepípede. La manera de fer el càlcul el podem veure a la taula 3.1.

Per altra banda, el mode atòmic ens mostrarà la forma atòmica dels diferents volums segons el tipus de material associat que tinguin. És a dir cada material tindrà una estructura atòmica diferent. Aquesta estructura estarà definida en una estructura de dades composta per 3 vectors base i un nombre indeterminat de vec-

Taula 3.1: Càlcul dels 8 vèrtex d'un paral·lelepípede a partir del seu vèrtex origen i la mida de les 3 dimensions

Donat un vèrtex origen (x,y,z) i les mides del paral·lelepípede (midax, miday, midaz)			
Nº vèrtex	**Component X del vèrtex**	**Component Y del vèrtex**	**Component Z del vèrtex**
1	x	y	z
2	x	y	z + midaz
3	x	y + miday	z
4	x	y + miday	z + midaz
5	x + midax	y	z
6	x + midax	y	z + midaz
7	x + midax	y + miday	z
8	x + midax	y + miday	z + midaz

tors d'offset. A partir dels 3 vectors base calcularem els centres dels àtoms de l'estructura aplicant el següent càlcul:

$x = origen.x + vectoratomic1.x * n + vectoratomic2.x * m + vectoratomic3.x * l$

$y = origen.y + vectoratomic1.y * n + vectoratomic2.y * m + vectoratomic3.y * l$

$z = origen.z + vectoratomic1.z * n + vectoratomic2.z * m + vectoratomic3.z * l$

$\forall l, m, n \in N$

Cal tenir en compte que haurem d'aturar-nos quan $x > origen.x * midax$ o bé $y > origen.y * miday$ o bé $z > origen.z * midaz$. De la mateixa manera els vectors d'offset ens permetran calcular els centres d'àtoms que s'han d'afegir a l'estructura, sense estar connectats amb la resta d'àtoms. Aquests centres els calcularem de la següent manera:

$x = origen.x + vectoratomic1.x * (n + offset_i) + vectoratomic2.x * (m + offset_i) + vectoratomic3.x * (l + offset_i)$

$y = origen.y + vectoratomic1.y * (n + offset_i) + vectoratomic2.y * (m + offset_i) + vectoratomic3.y * (l + offset_i)$

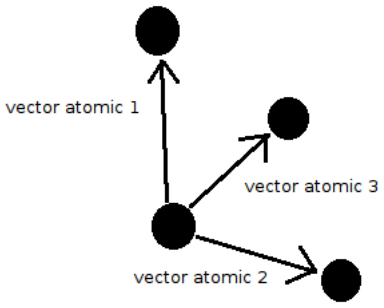
$z = origen.z + vectoratomic1.z * (n + offset_i) + vectoratomic2.z * (m + offset_i) + vectoratomic3.z * (l + offset_i)$

$\forall l, m, n \in N$

$i \in 1 : Noffs$ on Noffs és el nombre d'offsets.

Cal tenir en compte que haurem d'aturar-nos quan $x > origen.x * midax$ o bé $y > origen.y * miday$ o bé $z > origen.z * midaz$. Un cop calculats tots els centres dibuixarem les esferes que representaran l'àtom a partir d'aquests i amb un

Figura 3.5: Enllaços entre àtoms



radi concret que definirem segons la grandària que ens interessi. Per altra banda, caldrà dibuixar els enllaços entre àtoms utilitzant línies, que per tant requeriran 2 vèrtex per ser dibuixades. Aquestes línies utilitzaran els centres calculats amb els vectors atòmics base. Cada centre estarà connectat amb els 3 centres veïns que calcularem sumant cadascun dels 3 vectors atòmics base, tal i com es veu en la figura 3.5.

3.2.2 Anàlisi funcional de la comunicació nucli - interfície gràfica

Tenint en compte que un dels objectius que ens vam marcar inicialment, pel projecte, era el de reutilitzar totes les funcionalitats ja implementades en el nucli sense duplicar el codi, en aquesta part, el nostre principal problema era saber amb exactitud quines dades calia que transferíssim al nucli de simulació, fos com fos la comunicació, per tal que pogués realitzar la comprovació i simulació del disseny. El nucli de simulació requereix la següent informació per realitzar la comprovació de la validesa del disseny:

- El conjunt de volums que conformen el disseny amb les seves coordenades (x,y,z inicial i final), el nombre de malles en cada dimensió i la resta de característiques associades.

- El conjunt de materials juntament amb les seves característiques associades.

- El conjunt de propietats generals del projecte.

Amb aquests 3 conjunts el nucli ja serà capaç de comprovar la validesa del disseny i fer-ne la simulació.
Per altra banda cal analitzar en profunditat quin tipus de comunicació requerirà cadascun dels casos en que caldrà comunicació entre el nucli i la interfície. En primer lloc tenim el cas en que l'usuari voldrà comprovar que el disseny que està realitzant compleix amb les regles establertes per tal que la simulació pugui realitzar-se correctament. En aquest cas, caldrà una comunicació ràpida i eficaç, transparent a l'usuari, per tal que aquesta comprovació sigui una tasca trivial i no alenteixi el seu treball. En segon lloc tenim el cas de la realització de la simulació. En aquest cas, a causa de l'elevat temps de simulació i de la independència d'aquesta tasca de la interfície, no cal una integració tan gran. Simplement cal proporcionar les dades anteriorment esmentades d'una manera fàcilment interpretable pel nucli.

3.2.3 Anàlisi funcional de la part de configuració de la simulació

En aquest apartat, calia conèixer quin tipus de dades calia configurar. Tal i com s'ha dit anteriorment els paràmetres a configurar seran ampliables, per la qual cosa, l'aplicació els haurà de tractar de manera genèrica. L'únic que ens cal saber són les restriccions de tipatge dels paràmetres. Amb el client es va acordar la necessitat de suportar els següents tipus:

1. Booleà
2. Enter
3. Double

Per altra banda, pel que fa a la configuració de l'escala de les unitats, aquesta haurà de suportar com a unitat mínima l'Amstrong (10^{-10} metres) fins als mil·límetres. Finalment, pel que fa a la possibilitat del desenvolupador d'afegir

noves propietats o materials, el principal requeriment és que no calgui tocar el codi font de la interfície. Bàsicament volem que la interfície sigui autoprogramables per tal que s'adapti als canvis i ampliacions que rebi el nucli de simulació.

3.2.4 Anàlisi funcional de la part de guardar/carregar dissenys

Sobre aquesta funcionalitat el principal requeriment funcional és que el disseny que es guardi ha de ser compatible tan amb la posterior càrrega des de la interfície, com amb la càrrega des del nucli de simulació.

Capítol 4

Disseny i implementació software del projecte

En aquest capítol exposarem el disseny de la solució software que hem realitzat, per tal de complir els requeriments funcionals establerts i també els objectius del projecte.

4.1 Diagrama de classes UML

En primer lloc, farem una explicació del diagrama de classes UML (Unified Modeling Language) de la nostra aplicació des d'un punt de vista general per posteriorment entrar en detall en les decisions de disseny més importants i remarcables. El diagrama general el podeu veure a la figura 4.1 a la pàgina 37.

Com podem veure conté 3 classes relacionades amb les dades de l'aplicació:

1. Material
2. Property
3. Volum

Aquestes classes seran les que contindran les dades que l'usuari introduirà a través de la interfície i que posteriorment seran transmeses al nucli de simulació. A més

inclouran totes les operacions que es podrà fer sobre aquestes dades.

Per altra banda tenim 6 classes que són les que conformen la interfície gràfica:

1. MainWindow
2. GLWidget
3. MaterialsDialog
4. ScaleDialog
5. VolumsDialog
6. PropertiesDialog

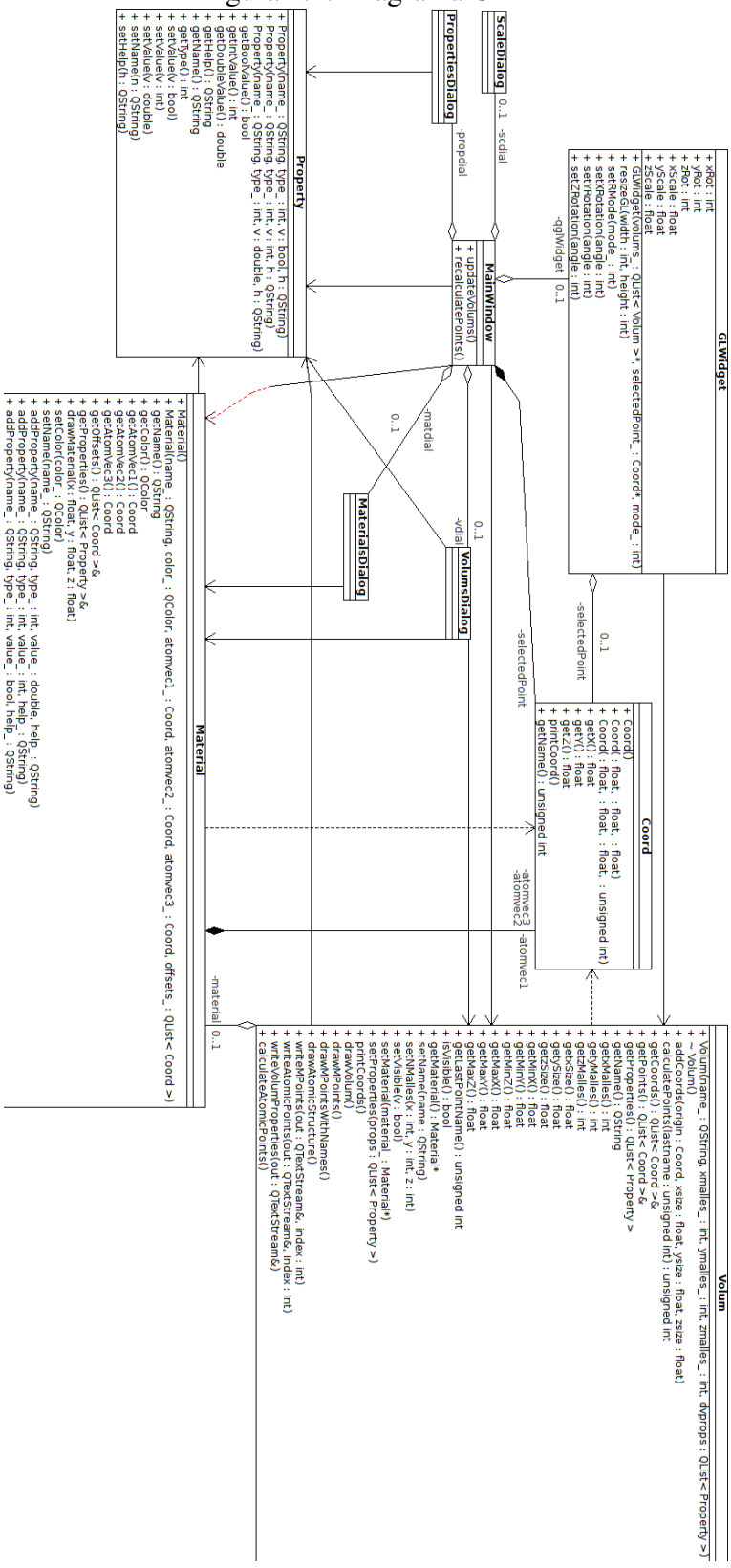
En primer lloc tenim la classe MainWindow, la qual és la classe que conté els diferents elements que conformen la finestra principal i les operacions que defineixen el seu comportament. A més, aquesta classe conté una instància de les 5 classes restants. Per una banda, inclou una instància del GLWidget, que és l'objecte gràfic que contindrà la representació 3D, feta amb OpenGL, de l'estructura electrònica. Per l'altra tindrà instàncies dels diferents diàlegs de configuració que estaran preparats per quan l'usuari els cridi. Finalment la classe MainWindow contindrà la llista de materials, la llista de volums i la llista de propietats generals del disseny els elements de les quals seran objectes de les 3 classes relacionades amb les dades que hem comentat inicialment. Els diferents diàlegs accediran a les dades que necessitin a través d'apuntadors per tal de que les dades mantinguin la integritat.

4.1.1 Classe Volum

Aquesta classe representa els volums que conformen l'estructura nanoelectrònica del disseny a simular. Conté la següent informació sobre el volum:

- El **nom del volum** el qual el posarà l'usuari per tal de tenir un identificador de cadascun.

Figura 4.1: Diagrama UML



- Les **mides del volum** en cadascun dels 3 eixos x,y,z.
- El **nombre de malles** que té el volum en cadascun dels 3 eixos x,y,z.
- Una variable booleana que ens indicarà si el volum és visible en aquell moment en la representació 3D.
- Una llista amb els **8 vèrtex que conformen el paral·lelepípede del volum**. Els vèrtex seran objectes de la classe Coord, la qual representa les coordenades 3D i un conjunt d'operacions que es pot realitzar amb elles. És una classe molt simple que no mereix més esment en aquesta memòria.
- Un **apuntador al material** (en concret a l'objecte de la classe Material) al qual està vinculat aquell volum, el qual ens permetrà accedir a les característiques o operacions que necessitem d'aquest.
- Una llista dels **vèrtex on es creuen les diferents malles**, per tal de poder fer la selecció de l'origen de nous volums que volem inserir.
- Una llista dels **centres dels diferents àtoms** que conformen l'estructura atòmica del volum, els quals ens serviran per realitzar el dibuixat d'aquesta quan calgui.
- Una llista de **propietats vinculades al volum** que l'usuari podrà modificar a través del diàleg de configuració corresponent. El elements d'aquesta llista seran objectes de la classe Property. Les propietats concretes associades al material vindran definides pels desenvolupadors i es carregaran a l'inici de l'aplicació amb uns valors per defecte que s'aplicaran a tots els volums.

La classe Volum, sobre aquestes dades, pot realitzar tot un seguit d'operacions, definides com a funcions de la classe. Volem destacar les següent per la seva rellevància:

- **void addCoords(Coord origin, float xsize, float ysize, float zsize)**: a partir del punt d'origen del volum i les seves mides calcula la resta de punts que conformen el paral·lelepípede, tal i com expliquem al capítol anterior. Aquesta funció serà utilitzada en el moment d'afegir un nou volum.

- **unsigned int calculatePoints(unsigned int lastname)**: aquesta funció ens permet calcular tots els punts on es creuen malles.

- **void calculateAtomicPoints()**: aquesta funció ens permet calcular el conjunt de centres de l'estructura atòmica del volum, ho farà a partir dels vectors que ens proporciona el material vinculat al volum de la manera com expliquem al capítol anterior.

- **void drawVolum()**: aquesta funció permet dibuixar el volum en mode sòlid al widget OpenGL.

- **void drawAtomicStructure()**: aquesta funció permet dibuixar el volum en mode atòmic al widget OpenGL.

4.1.2 Classe Material

Aquesta classe representa els materials que l'usuari tindrà disponibles per assignar-los als volums. Cada volum tindrà vinculat un únic material. Els materials i els seus atributs vindran definits pels desenvolupadors a través d'un fitxer de text el qual l'aplicació llegirà a la seva inicialització. La classe conté els següents atributs:

- El **nom del material** per tal de poder identificar-lo.

- Els **3 vectors atòmics** que ens permeten calcular els centres atòmics tal hi com hem explicat al capítol anterior.

- El **vectors d'offset** que ens permeten calcular els centres dels àtoms que es situen enmig de l'estructura atòmic sense enllaços.

- El **color** que volem que tingui el volum d'aquell material en la representació 3D.

- Una llista de **propietats vinculades al material** que l'usuari podrà configurar. Els elements d'aquesta llista seran objectes de la classe Property.

La classe Material, sobre aquestes dades, pot realitzar tot un seguit d'operacions, definides com a funcions de la classe. Volem destacar les següent per la seva rellevància:

- **void drawMaterial(float x, float y, float z)**: aquesta funció és l'encarregada de dibuixar els àtoms del material. És cridada des de la funció drawAtomicStructure() de la classe Volum.

4.1.3 Classe Property

Aquesta és una de les classes més importants pel seu disseny que permet als desenvolupadors del nucli del simulador que les diferents propietats vinculades tan al projecte en general, als materials o bé als volums siguin modificables sempre que vulguin. Això els ofereix una flexibilitat molt gran, cosa que els permet centrar-se en el treball en el nucli, sense haver-se de preocupar de l'adaptació de la interfície gràfica. Aquestes propietats estaran definides en fitxers de text i es carregaran a l'inici creant objectes d'aquesta classe. Els atributs de la classe són:

- El **nom de la propietat**, el qual ens permetrà identificar-la.
- El **tipus de la propietat**, el qual serà un enter. En cas de ser 0 es referirà a que la propietat és un booleà, si és 1 un enter i si és un 2 un double.
- El **valor de la propietat**, el qual serà un double i s'interpretarà sempre segons el seu tipus indicat a l'atribut anterior.
- La **cadena d'ajuda**, útil perquè l'usuari tingui una descripció més detallada de la propietat.

La classe en sí no té operacions destacables més enllà de les estàndard per accedir i definir aquests atributs.

4.1.4 Classe MainWindow

Aquesta classe determina el comportament de la finestra principal segons els esdeveniments que produeix l'usuari sobre aquesta i els seus elements. La definició

dels elements que conté i la posició on es situen es fa a través de l'eina Qt Designer que ens permet dissenyar la interfície gràfica de manera fàcil i eficient. Aquesta eina genera una classe que conté únicament els elements de la interfície i un objecte de la qual serà un atribut més de la MainWindow, que s'inicialitzarà en el constructor d'aquesta. El disseny de la interfície de la finestra s'explicarà en la següent secció. La resta d'atributs de la classe són:

- La **llista de volums** que conformen el disseny. Els elements de la llista seran objectes de la classe Volum.
- La **llista de propietats** generals de la simulació. Els elements de la llista seran objectes de la classe Property. L'usuari podrà configurar-les a través del diàleg corresponent.
- La **llista de propietats per defecte dels volums** que ens serviran per inicialitzar nous objectes d'aquest tipus. Aquestes propietats es carregaran a l'inici de l'aplicació llegint-los del fitxer de text corresponent.
- La **llista dels materials** disponibles per assignar als materials. Els elements de la llista seran objectes de la classe Material i es carregaran a l'inici de l'aplicació llegint-los del fitxer de text corresponent.
- El **punt seleccionat**. Per tal de facilitar la inserció de nous volums, el widget OpenGL permetrà que seleccionem un dels punts on es creuen les malles de qualsevol dels volums existents. Aquest punt s'utilitzarà després com a origen a l'afegir un nou volum.
- El **fitxer actual** el qual és una cadena que ens indica el fitxer al qual es guardarà el projecte. Aquesta variable estarà inicialitzada sempre que haguem obert un disseny o bé haguem guardat, com a mínim un cop un nou disseny.
- L'**escala** que s'aplicarà al conjunt del disseny. Aquest atribut s'utilitzarà al desar el disseny i es multiplicarà a totes les unitats espacials relacionades amb els volums. Permet a l'usuari definir amb quines unitats està definit el disseny.

La classe MainWindow, sobre aquestes dades, pot realitzar tot un seguit d'operacions, definides com a funcions de la classe. Volem destacar les següent per la seva rellevància:

- **void updateVolums()**: Aquesta funció es cridada sempre que s'afegeix un nou volum i s'encarrega d'actualitzar els elements de la finestra que es veuen afectat per aquest canvi. En particular el widget que llista els volums existents en el disseny i el widget OpenGL que té la seva representació 3D.
- **bool saveFile(const QString &fileName)**: Aquesta funció permet desar el disseny actual en el fitxer que li indiquem.
- **void loadFile(const QString &fileName)**: Aquesta funció permet carregar un disseny des del fitxer que li indiquem.
- **void loadMaterials()**: Aquesta funció carrega els diversos materials llegint els fitxers que hi ha la carpeta Materials. Cada fitxer correspondrà a material. La funció serà cridada a l'inici de l'aplicació.
- **void loadProperties()**: Aquesta funció carrega les propietats generals per defecte del disseny des del fitxer properties.txt. La funció serà cridada a l'inici de l'aplicació.
- **void loadDefaultVolumProperties()**: Aquesta funció carrega les propietats per defecte vinculades als volums des del fitxer volum_properties.txt. La funció serà cridada a l'inici de l'aplicació.

A més, hi ha tot un seguit de funcions que permeten controlar el comportament dels elements de la interfície, que s'activen segons els esdeveniments que aquests generen. Per altra banda, també hi ha funcions que carreguen diàlegs de configuració com el de l'escala, el de les propietats generals o el dels materials.

4.1.5 Classe GLWidget

Aquesta classe és la que ens permetrà incrustar un widget a la finestra principal per a la representació 3D de les nanoestructures electròniques a través de l'ús de

la tecnologia OpenGL. Aquesta classe, hereda de la classe de Qt QGLWidget que és la que proporciona integració amb OpenGL. L'únic que cal és reimplementar un seguit de funcions (paintGL(), resizeGL() i initializeGL()) de la classe pare per tal de tenir en funcionament el widget amb tota la potencialitat de l'OpenGL. Els atributs de la classe són:

- L'**escala de l'escena** en els tres eixos x,y,z.
- L'**angle de rotació l'escena** en els tres eixos x,y,z.
- Un apuntador a **la llista de volums** per poder accedir a ells i dibuixar-los en l'escena.
- Un atribut enter que ens indica en quin **mode de visualització** està el widget.
- Un apuntador a **la coordenada del vèrtex seleccionat**.

Les funcions que cal destacar de la classe són les següents:

- **void initializeGL()**: Aquesta funció ens permet inicialitzar l'entorn OpenGL, bàsicament s'estableix el color de fons i s'habiliten els efectes i llums que necessitem.
- **void paintGL()**: Aquesta funció és l'encarregada de pintar l'escena. Establirà l'escala i la rotació de l'escena per posteriorment passar a dibuixar tots els volums del disseny, tenint en compte el mode de visualització en el què estem.
- **void resizeGL(int width, int height)**: Aquesta funció s'activa automàticament quan hi ha un reamidament de la finestra, per tal d'adaptar l'escena a les noves mides.
- **void keyPressEvent(QKeyEvent *event)**: Aquesta funció controla els esdeveniments de teclat. Sempre que es prem una tecla d'aquest es crida automàticament. Dins d'aquesta funció establim quin comportament volem

segons la tecla pitjada. Per exemple, definirem que amb els cursors podem moure l'escena.

- **void mousePressEvent(QMouseEvent *event)**: Aquesta funció controla els esdeveniments dels clics del ratolí. Sempre que fem un clic es cridarà automàticament. En aquesta funció, en combinació amb la següent, és on programarem la rotació de l'escena a través del ratolí amb el botó esquerra apretat i movent-lo.

- **void mouseMoveEvent(QMouseEvent *event)**: Aquesta funció s'activa automàticament quan hi ha un moviment del ratolí. Com hem dit combinada amb l'anterior aconseguim rotar l'escena a través del ratolí. El funcionament és el següent: L'usuari fa clic esquerra sobre el widget. La funció anterior desa el punt on s'ha fet el clic. Posteriorment s'arrossega el ratolí, la funció va calculant la diferència entre el punt inicial i l'actual i aplica la rotació de la pantalla segons aquesta diferència.

- **void wheelEvent(QWheelEvent *event)**: Aquesta funció s'activa automàticament quan hi ha un moviment de la rodeta del ratolí. En aquesta funció implementarem el zoom a l'escena. Sí movem la rodeta cap amunt s'incrementarà el coeficient d'escala de l'escena en un 10%, en cas contrari el disminuirem un 10%.

- **Coord SelectElement(const unsigned int cursorX, const unsigned int cursorY)**: Aquesta funció serà la que ens permetrà esbrinar si l'usuari ha clicat algun dels vèrtex on es creuen les malles dels volums, per establir-lo com a punt d'origen. Es cridarà des de la funció que gestiona els esdeveniments de clic del ratolí (mousePressEvent).

4.1.6 Classes dels diàlegs de configuració

Com podem veure al diagrama de classes UML a més de les classes que hem explicat, n'hi ha quatre més que corresponent als diàlegs de configuració de l'aplicació. Aquests diàlegs, com hem dit seran cridats per part de la MainWindow

quan l'usuari a través del menú que inclou aquesta seleccioni l'opció corresponent, o bé, en el cas del diàleg VolumsDialog, l'usuari hi accedirà a través dels butons d'afegir o editar volums. Aquests diàlegs hereten de la classe de Qt QDialog. Aquesta classe permet que quan s'iniciïn els diàlegs la finestra principal quedi bloquejada i l'usuari només pugui interactuar amb aquests. Diem que el diàleg està en mode modal. Si ho desitgéssim també podríem obrir-lo en mode no modal de manera que poguéssim interactuar amb les dues finestres, però en el nostre cas no ens interessa, ja que generaria problemes en la integritat de les dades. Cal dir que aquestes classes, de la mateixa manera que la MainWindow, només defineixen el comportament del diàleg segons els esdeveniments que produeix l'usuari. Els elements de la interfície venen definits d'idèntica forma que en la MainWindow. El disseny d'aquests s'explicarà a la següent secció. Seguidament farem una petita explicació de les quatre classes:

1. **Classe MaterialsDialog**: Aquesta classe permetrà a l'usuari configurar els diferents paràmetres que té cada material. Contindrà una llista dels materials existents i una llista de les propietats que podem configurar. Un cop seleccionat el material i la propietats l'usuari podrà especificar el seu valor. Segons el tipus d'aquesta propietat se li presentarà un widget diferent adequat pel tipus corresponent. Per exemple, si és un booleà, tindrem un menú desplegable amb els valors true i false.

2. **Classe PropertiesDialog**: Aquesta classe permetrà a l'usuari configurar les propietats generals de la simulació. Contindrà una llista de les propietats que l'usuari pot configurar i en seleccionar-ne una el funcionament serà igual que en el cas anterior.

3. **Classe ScaleDialog**: Aquesta classe permetrà a l'usuari establir l'escala que s'aplicarà al disseny. Podrà definir les unitats (des de Amstrongs fins a mil·límetres) i un valor d'aquestes. Les mesures espacials finals seran la multiplicació de la mesura, per el valor de la unitat d'escala, per la unitat d'escala.

4. **Classe VolumsDialog**: Aquesta classe ens permetrà tenir un diàleg per a

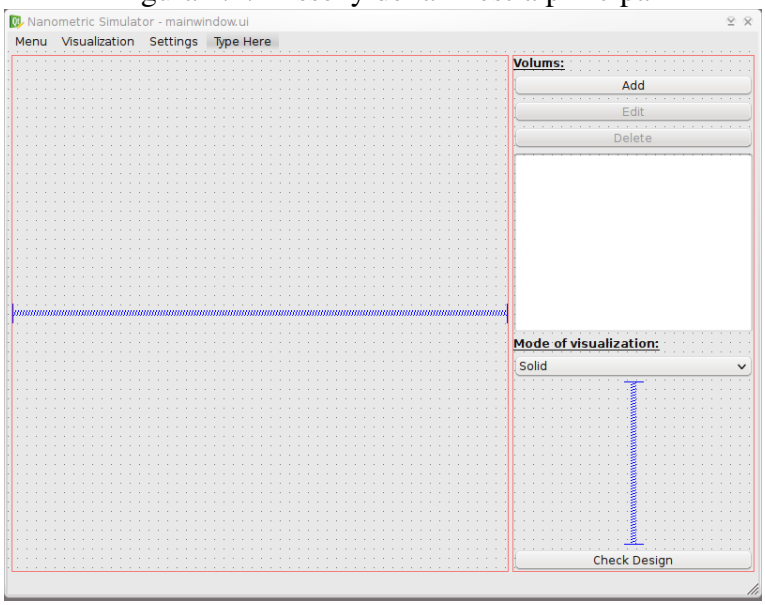
la inserció o edició de volums. Tindrà un constructor per cadascuna de les dues possibilitats. En cas d'afegir un nou volum l'usuari haurà d'omplir les dades necessàries (punt d'origen, mides del volum, nombre de malles, etc.). En cas d'editar un volum el diàleg carregarà les seves dades i l'usuari podrà modificar les que vulgui. En els dos casos, al acceptar els canvis el diàleg comprovarà que les dades són vàlides (que el volum té un nom no duplicat i que no es solapa amb cap altre volum a nivell espacial). Si tot és correcta, s'efectuaran els canvis a les dades, sinó, es tornarà a mostrar el diàleg, perquè l'usuari ho corregeixi.

4.2 Disseny de la interfície

En aquesta secció ens centrarem en el disseny de la interfície de la nostra aplicació. Com ja hem esmentat, el disseny s'ha realitzat a través de la tecnologia Designer de Qt, que et permet fer-ho de manera visual, sense haver de picar cap línia de codi. Un cop tens el disseny acabat, aquest es desa en un fitxer .ui que el precompilador de Qt és capaç de transformar en una classe C++ la qual, podem utilitzar allà on necessitem. El disseny de les diferents finestres s'ha fet pensant en facilitar l'usuari l'ús de la interfície. Els principals criteris seguits han estat el de proporcionar les funcionalitats més habituals en el mínim de clics i el de mantenir una estructura intuïtiva. Seguidament us mostrarem el disseny de la finestra principal i dels quatre diàlegs de configuració:

1. **Finestra principal**: En la figura 4.2 podem veure el disseny de la finestra principal. En el menú superior podrem accedir a les funcionalitats de desar i carregar disseny i a la de configurar els materials, propietats i escala. A la part central observem un buit que és el que omplirà el widget OpenGL. Al ser un widget personalitzat no el podem afegir a través del Designer i per això li reservem un espai; en el constructor de la classe serà on li inserterem. Finalment a la barra lateral dreta tenim tot el relacionat amb els volums (la llista, i els botons per afegir, editar i eliminar), la selecció del mode de visualització, i el botó per realitzar la comprovació del disseny.

Figura 4.2: Disseny de la finestra principal



Com podem veure les funcionalitats estan agrupades per tipologia per fer intuïtiva la interfície i a més les funcionalitats relacionades amb els volums són les més accessibles, perquè són les més utilitzades.

2. **Diàleg de configuració dels materials**: En la figura 4.3 podem veure el disseny del diàleg de configuració dels materials. Com podem veure a la part esquerra superior es podrà seleccionar el material a configurar. Un cop seleccionat, a sota es llistaran les propietats. Finalment al seleccionar-ne una, a la part dreta, ens apareixerà el widget per tal de configurar-la amb el valor actual. Segons el tipus de la propietat el widget serà un o un altre. Com podem veure el disseny del diàleg és intuïtiu ja que la distribució dels elements segueix l'ordre en que s'han de realitzar els diferents passos.

3. **Diàleg de configuració de les propietats generals**: En la figura 4.4 podem veure el disseny del diàleg de configuració de les propietats generals. Com podem observar, el diàleg és molt semblant al de materials. Amb l'única diferència que no hi ha desplegable per escollir el material. Per tan el funcionament és idèntic exceptuant aquest punt.

Figura 4.3: Disseny del diàleg de configuració dels materials

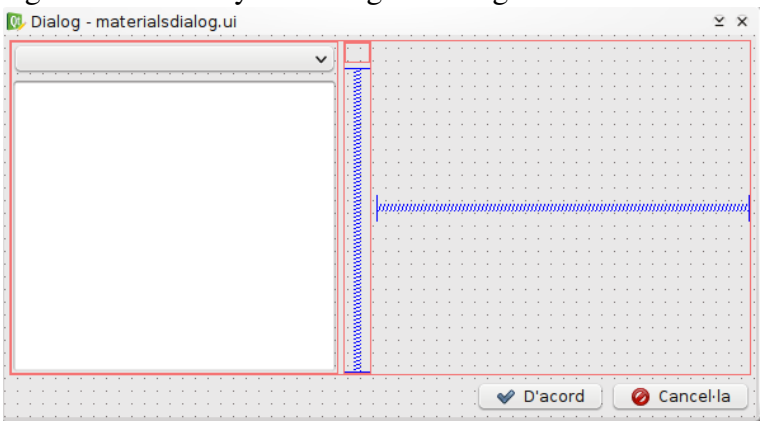


Figura 4.4: Disseny del diàleg de configuració de les propietats generals del disseny

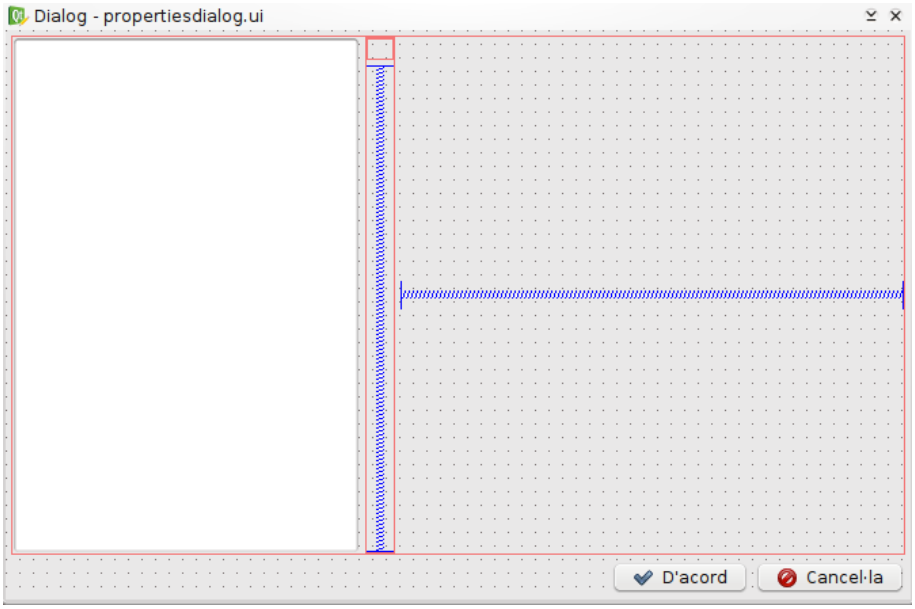
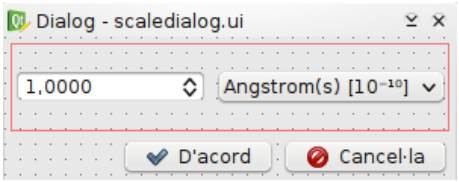


Figura 4.5: Disseny del diàleg de configuració de l'escala del disseny



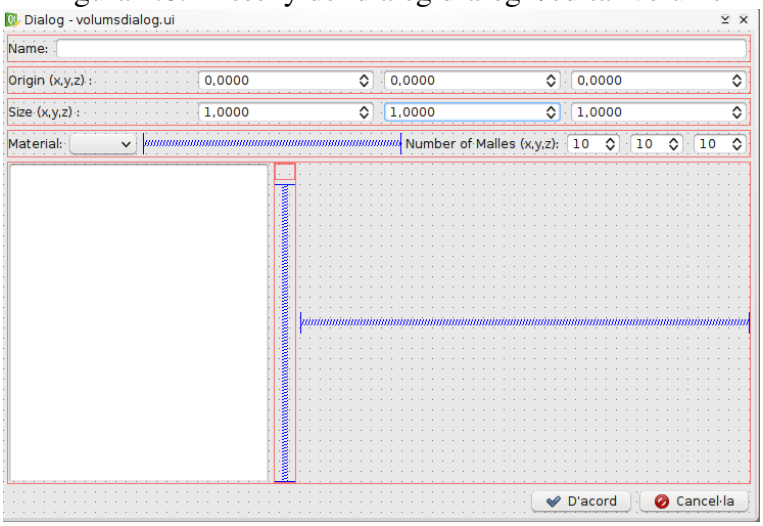
4. **Diàleg de configuració de l'escala**: En la figura 4.5 podem veure el disseny del diàleg de configuració de l'escala. Com podem observar és un diàleg molt simple que consta d'un boto de selecció del valor de l'escala i un desplegable per escollir les unitats.

5. **Diàleg d'afegir/editar volums**: En la figura 4.6 podem veure el disseny del diàleg d'afegir/editar volums. Com podem observar a la part superior hi ha tots els camps necessaris per configurar el volum. En cas d'afegir un nou volum el diàleg ens apareixerà tal i com es mostra a la figura. En cas d'editar un volum el diàleg ens apareixerà amb les dades d'aquell volum. Finalment la part inferior del diàleg és idèntica a la de configuració de propietats generals i funciona de la mateixa manera. L'estructura dels elements també té una distribució força intuïtiva malgrat en aquest cas tampoc podríem definir un ordre concret lògic. S'han ordenat els elements per importància. A més, la interfície permet canviar d'un element a l'altre a través del tabulador.

4.3 Estructures de dades per la comunicació nucli-interfície

En aquesta secció explicarem les estructures de dades necessàries per a la comunicació entre el nucli de simulació i la interfície gràfica. Aquestes estructures de dades són molt importants, com veurem, a causa de la decisió presa de realitzar la comprovació del disseny utilitzant el codi ja realitzat pel nucli de simulació. Cridarem la funció de comprovació adequada des de la nostra aplicació, com a funció externa. Això ho aconseguirem gràcies a la capacitat del compilador gcc

Figura 4.6: Disseny del diàleg d'afegir/editar volums



d'enllaçar codi FORTRAN amb codi C++ [28]. Aquesta funció externa feta amb FORTRAN serà cridada quan cliquem el botó de comprovació del disseny i li passarem les dades necessàries utilitzant unes estructures de dades compartides per la nostra aplicació i la funció. El fet d'usar una funció externa fa que haguem de declarar les estructures tan en la nostra aplicació com en el codi FORTRAN. Cal anar amb compte de declarar les estructures amb un format exactament igual i equivalent. En realitat el que li passem a la funció són apuntadors a les estructures. I ella no les sabria identificar si no fos perquè a dins la funció li estem definint quin és el tipus de dades d'aquell apuntador. És per això que és tan important que coincideixin perfectament. Un petit canvi d'ordre fa que que la funció accedeixi als blocs de memòria de manera errònia. Per altra banda, cal destacar que és necessari posar les dades per ordre de grandària per facilitar l'alineament en FORTRAN. Aplicats aquests criteris a la taula 4.1 podeu veure les estructures dissenyades tan en un llenguatge com en l'altre. Com podeu veure l'estructura de propietats es fa servir com a element de les altres dues estructures per tal de que tan els volums com els materials tinguin la opció de que els desenvolupadors del nucli adaptin el nombre de propietats a les seves necessitats. Per limitacions del llenguatge FORTRAN i pel fet d'haver de tenir aquesta estructura tan rígida de dades, s'ha hagut de declarar els vectors de propietats interns a les estructures

Taula 4.1: Estructures utilitzades per la comunicació nucli-interfície en C++ i FORTRAN

	Estructures C++	**Estructures FORTRAN**

Estructura pels paràmetres	`struct parameter_str {`
	`  char help[300];`
	`  char nom[20];`
	`  double valor;`
	`  int tipus;`
	`};`
		`TYPE DEFINICIOPARAMETRES`
		`  CHARACTER*300 HELP`
		`  CHARACTER*20 NOM`
		`  DOUBLE PRECISION VALOR`
		`  INTEGER TIPUS`
		`END TYPE DEFINICIOPARAMETRES`

Estructura pels volums	`struct volum_str {`
	`  parameter_str params[MAXPARAM];`
	`  char material[20];`
	`  double xi,yi,zi;`
	`  double xf,yf,zf;`
	`  int mxi, myi, mzi;`
	`  int mxf, myf, mzf;`
	`};`
		`TYPE DEFINICIOVOLUM2`
		`  TYPE (DEFINICIOPARAMETRES)`
		`  :: PARAMETRES(1:MAXPARAM)`
		`  CHARACTER*20 MATERIAL`
		`  DOUBLE PRECISION XI`
		`  DOUBLE PRECISION YI`
		`  DOUBLE PRECISION ZI`
		`  DOUBLE PRECISION XF`
		`  DOUBLE PRECISION YF`
		`  DOUBLE PRECISION ZF`
		`  INTEGER MXI`
		`  INTEGER MYI`
		`  INTEGER MZI`
		`  INTEGER MXF`
		`  INTEGER MYF`
		`  INTEGER MZF`
		`END TYPE DEFINICIOVOLUM2`

Estructura pels materials	`struct material_str {`
	`  parameter_str params[MAXPARAM];`
	`  char nom[20];`
	`};`
		`TYPE DEFINICIOMATERIAL2`
		`  TYPE (DEFINICIOPARAMETRES)`
		`  :: PARAMETRES(1:MAXPARAM)`
		`  CHARACTER*20 NOM`
		`END TYPE DEFINICIOMATERIAL2`

de volums i materials amb una mida estàtica amb el nombre d'elements màxim definit a MAXPARAMS. El problema és que a la funció FORTRAN estem definint el paràmetre que ens passaran i no ens permet que aquest sigui una estructura amb un element intern de mida variable en temps de compilació. En tot cas, no ha de ser un problema ja que donem l'opció de que hi hagi 50 paràmetres definits pels desenvolupadors del nucli, amb opció a ser ampliat molt fàcilment si ho desitgen.

4.4 Fitxers d'entrada i sortida

Tal i com s'ha anat veient, la interfície requerirà de la presència de tot un seguit de fitxers amb dades d'entrada (que llegirà en la seva inicialització) per tal de poder funcionar. Aquestes dades vindran proporcionades pels desenvolupadors del nucli de simulació i els permetrà adaptar la interfície a les seves necessitats sense haver de modificar-ne el codi. A més a més, tindrem un fitxer de sortida que tindrà 2 objectius molt clars:

1. En primer lloc, ens servirà per guardar els dissenys realitzats, per poder-los modificar sempre que volguem a través de la nostra aplicació. O bé per fer-ne de nous basant-nos en algun dels existents i estalviar-nos feina.

2. Per altra banda, aquest fitxer també serà l'entrada pel nucli de simulació. Contindrà les dades necessàries del disseny per tal que el nucli pugui realitzar la simulació.

En el cas de la comprovació del disseny, la qual es realitza cridant una funció externa en FORTRAN que forma part del codi del nucli, no necessitàvem fitxers per la necessitat de fer el màxim de ràpida i integrada amb la interfície aquesta comprovació. Això fa que la tasca del disseny sigui àgil, ja que la comprovació es realitzarà diverses vegades cada disseny. En canvi, en el cas de la simulació realitzada pel nucli, al ser un càlcul complex que triga hores i fins i tot dies, no és necessari aquest nivell d'integració amb la interfície. A més, el fet que normalment es faci córrer sobre clusters de càlcul externs al ordinador on es fa el disseny fa necessària la creació d'aquest fitxer d'entrada pel nucli amb les dades de la simulació.
Seguidament passarem a analitzar l'estructura que hem dissenyat tan pels fitxers d'entrada com pels de sortida:

1. A la taula 4.2 tenim l'estructura del fitxer de material. Cada material està especificat per un fitxer d'aquest tipus i tots ells estan situats a la carpeta Materials. Com podem veure, comença per una lletra M per tal d'assegurar-nos que el fitxer és de materials. En cas de no tenir aquesta lletra a l'inici

serà ignorat. Seguidament tenim el nom del material que serà una cadena de text, els 3 vectors (de 3 components x,y,z) per calcular els centres atòmics, el nombre de vectors d'offset i seguidament el llistat d'ells. A continuació hi tindrem el color en components RGB, dada que utilitzarà la interfície per mostrar els volums amb aquell material del color especificat. Finalment hi figuraran les propietats vinculades al material.

2. A la taula 4.3 hi podem veure l'estructura que servirà tan pel fitxer de propietats generals com pel fitxer de propietats dels volums. És una estructura simple on a l'inici s'indica el nombre de propietats que trobarem en el fitxer i seguidament totes elles seguint el format nom de la propietat, tipus, valor i cadena d'ajuda.

3. A la taula 4.4 tenim l'estructura del fitxer de sortida que generarà la interfície per desar els dissenys realitzats. Primerament contindrà una lletra V per comprovar que és un fitxer de disseny, en cas contrari l'aplicació ens informarà que el fitxer no té un format correcta. Seguidament hi trobarem l'escala del disseny que caldrà multiplicar a tots els valors espacials. A continuació hi tenim el nombre de volums que conté el fitxer, i a partir d'aquest ja podrem anar llegint-los tots. Per cada volum tindrem especificats el seu nom en una cadena de text, el nombre de malles en cada eix, el nom del material que té vinculat, les coordenades del seu vèrtex origen i finalment la mida en cadascun dels eixos.

Taula 4.2: Estructura del fitxer de materials

```
M
NOM_MATERIAL
VECTOR_1
VECTOR_2
VECTOR_3
NUMERO_OFFSETS
VECTOR_OFFSET_1
VECTOR_OFFSET_2
...
VECTOR_OFFSET_N
COLOR_RGB
NUMERO_PROPIETATS
NOM_PROPIETAT_1 TIPUS VALOR HELP
NOM_PROPIETAT_2 TIPUS VALOR HELP
...
NOM_PROPIETAT_N TIPUS VALOR HELP
```

Taula 4.3: Estructura del fitxer de propietats generals i dels volums

```
NUMERO_PROPIETATS
NOM_PROPIETAT_1 TIPUS VALOR HELP
NOM_PROPIETAT_2 TIPUS VALOR HELP
...
NOM_PROPIETAT_N TIPUS VALOR HELP
```

Taula 4.4: Estructura del fitxer del disseny

```
V ESCALA NUMERO_VOLUMS
NOM_VOLUM_1
NUMERO_MALLES_X NUMERO_MALLES_Y NUMERO_MALLES_Z
MATERIAL_VOLUM_1
VERTEX_ORIGEN_VOLUM_1
MIDA_EIX_X MIDA_EIX_Y MIDA_EIX_Z
NOM_VOLUM_2
NUMERO_MALLES_X NUMERO_MALLES_Y NUMERO_MALLES_Z
MATERIAL_VOLUM_2
VERTEX_ORIGEN_VOLUM_2
MIDA_EIX_X MIDA_EIX_Y MIDA_EIX_Z
...
NOM_VOLUM_N
NUMERO_MALLES_X NUMERO_MALLES_Y NUMERO_MALLES_Z
MATERIAL_VOLUM_N
VERTEX_ORIGEN_VOLUM_N
MIDA_EIX_X MIDA_EIX_Y MIDA_EIX_Z
```

Capítol 5

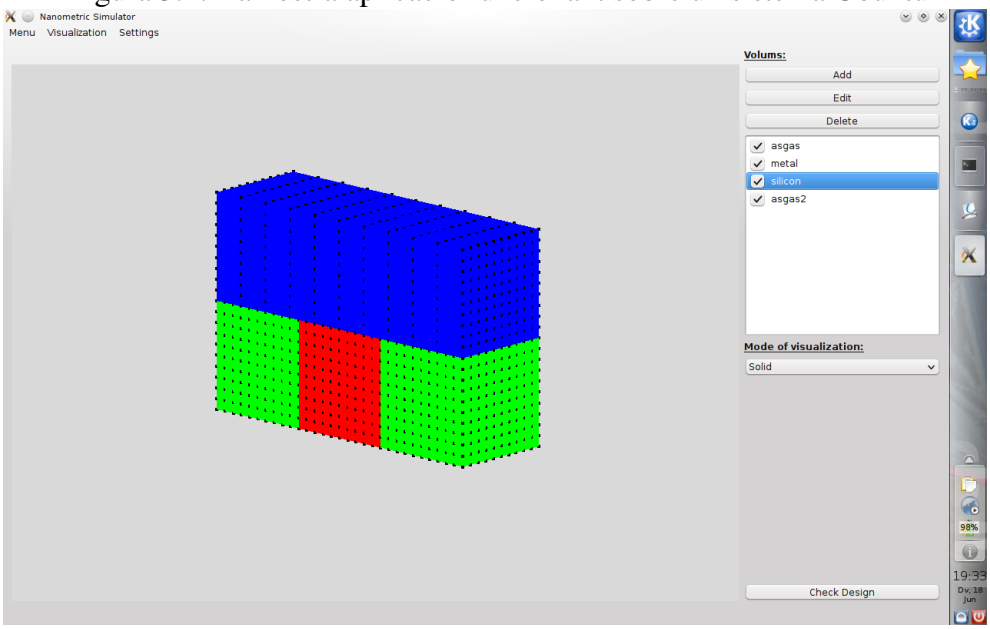
Tests

En aquest capítol comentarem els diferents testos que hem realitzat durant el desenvolupament del projecte per tal de comprovar el seu correcta funcionament.

5.1 Test de multiplataforma

Un dels primers testos que vam realitzar fou el de comprovar la correcta compilació i funcionament en els sistemes operatius GNU/Linux i Microsoft Windows (tan en la versió XP com en la versió 7). El sistema operatiu amb el què he desenvolupat l'aplicació ha estat sempre un sistema GNU/Linux, per tant el correcta funcionament en aquest sistema estava assegurat. Per altra banda, un cop finalitzada la codificació de la part 3D es va comprovar la correcta compilació i funcionament sobre un sistema Windows XP. En principi m'esperava algun tipus de problema, sobretot en la compilació, ja que el llenguatge C++ no està dissenyat per ser multiplataforma com el JAVA. Sorprenentment, cal dir que el nivell de portabilitat de l'aplicació fou perfecte i no va caldre modificar cap línia de codi. L'única complicació va sorgir al tornar a realitzar la prova un cop implementada la comunicació nucli-interfície. Llavors fou necessari preparar l'entorn de compilació per poder compilar codi FORTRAN. L'entorn de desenvolupament de Qt per Windows duu inclòs el gcc, però no el plugin de gcc per compilar fortran. Tot i així un cop instal·lat el plugin la compilació va funcionar a la perfecció i de nou

Figura 5.1: La nostra aplicació funcionant sobre un sistema Ubuntu



el comportament de l'aplicació fou l'esperat. En les figures 5.1, 5.2 i 5.3 podeu veure l'aplicació funcionant sobre els sistemes Ubuntu GNU/Linux, Microsoft Windows XP i Microsoft Windows 7 respectivament.

5.2 Test de la comunicació nucli-interfície

El desenvolupament d'aquesta part, fou potser, la que va comportar més problemes. A més, va caldre una estreta col·laboració amb els desenvolupadors del nucli. Es va començar amb un test simple en que des de la nostra aplicació cridàvem una funció simple de FORTRAN que ens omplia els paràmetres amb un valor i el mostràvem en una finestra emergent. Un cop comprovat que era possible tenir funcions externes fetes en FORTRAN vam passar a cridar estructures més complexes per veure com havíem de declarar-les a banda i banda. Finalment, un cop ja teníem l'estructura dissenyada vam fer proves serioses comprovant la correcte transferència de dissenys realitzats a través de la interfície a la funció de comprovació. Se li van passar tan dissenys correctes com dissenys amb algun tipus d'errada per tal de veure el correcta funcionament del sistema de comunicació.

Figura 5.2: La nostra aplicació funcionant sobre un sistema WindowsXP

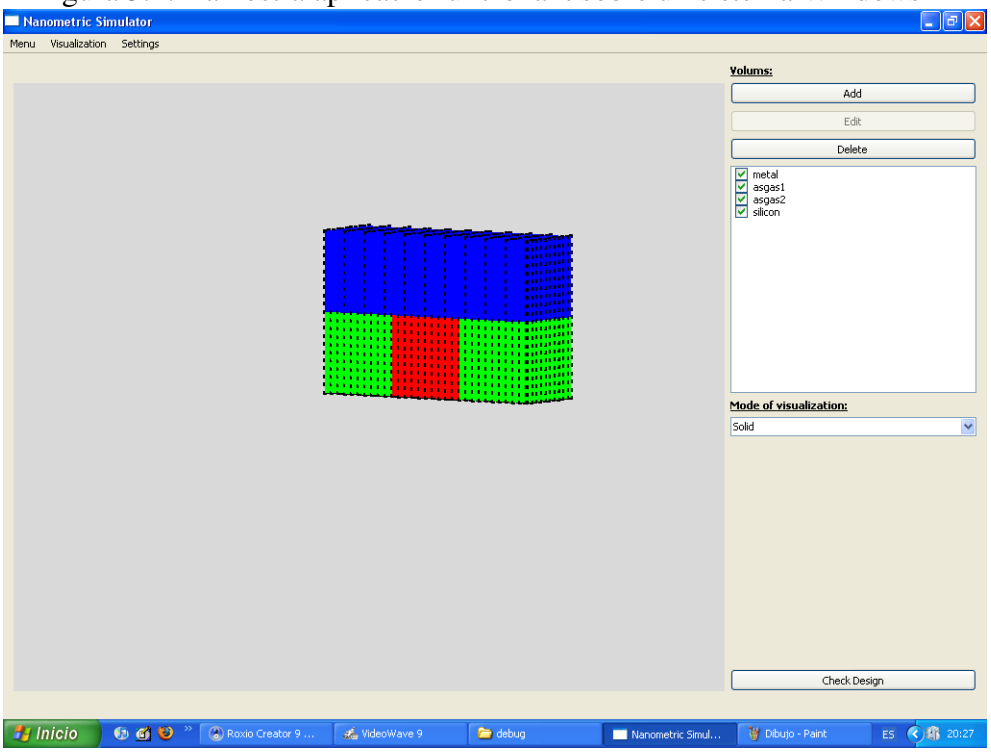
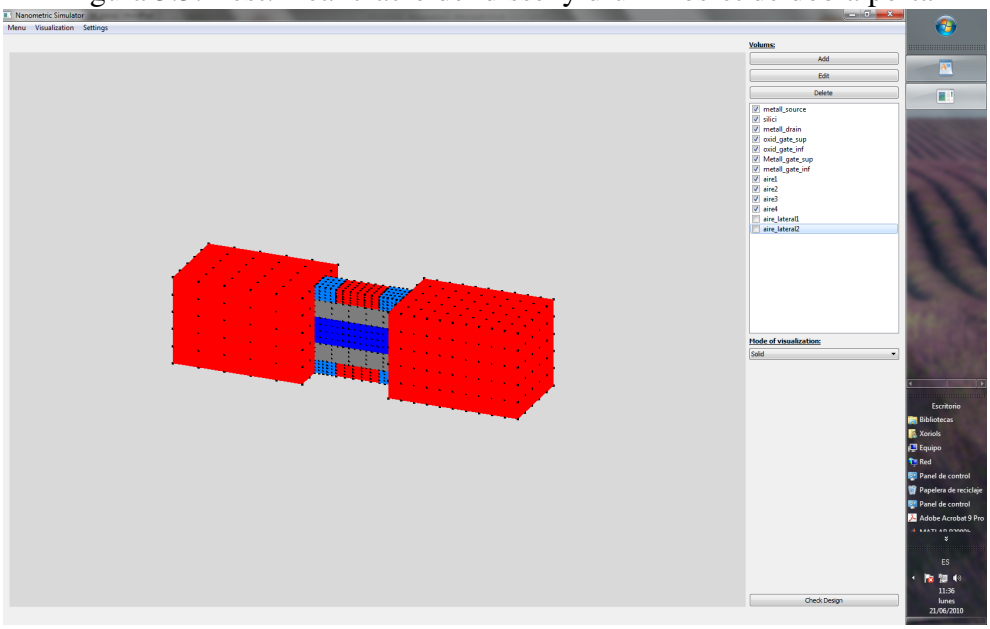


Figura 5.3: Test: Realització del disseny d'un Mosfet de dobla porta



5.3 Test de realització d'un disseny real

Un cop finalitzades les funcionalitats de l'aplicació es va passar a testejar-la intensament amb la realització de dissenys reals que ja s'havien fet anteriorment sense disposar de la interfície. Això, a part de permetre'ns corregir possibles errors, ens permetia fer una comparativa entre el temps i comoditat de fer un disseny amb la interfície o sense. Vam realitzar el disseny d'un transistor Mosfet de doble porta de 20 nm de longitud de canal i 4 nm d'òxid. Aquest transistor és un clar exemple del tipus d'estructures que es dissenyaran ja que segons el ITRS [2], els transistors de doble porta son els candidats idonis per els propers 5-10 anys. Això és degut al fet que la doble porta permet tenir un millor control electrostàtic de la conducció en el canal. Podeu veure el disseny d'aquest transistor a la figura 5.3.

El disseny d'aquest transistor abans de disposar de la interfície gràfica tenia una durada aproximada d'una jornada de treball (8 hores). El temps esmerçat utilitzant la nostra aplicació, tenint en compte que era el primer disseny que es feia, ha estat inferior a una hora de treball. Per tant, podem concloure que la interfície desenvolupada serà d'una gran utilitat i millorarà la productivitat dels

futurs usuaris del simulador.

5.4 Test de desat i càrrega d'un disseny real

Un últim test fou el de comprovar el correcta desat i càrrega d'un disseny real. Aprofitant el disseny fet en el test anterior, es va desar. Després es va reiniciar l'aplicació i es va carregar el disseny. Un cop fet això es va comprovar que el disseny era l'esperat i que les dades de les diferents propietats generals, dels diferents volums i dels diferents materials eren els correctes, així com el valor d'escala.

Capítol 6

Conclusions

Arribats a aquest punt amb el projecte ja dissenyat i implementats és moment de fer un anàlisi de la feina realitzada i pensar cap on podria avançar el projecte en un futur.

6.1 Assoliment dels objectius

Quan finalitza un projecte, per tal d'analitzar la feina feta, el què cal bàsicament, és centrar-nos en l'acompliment dels objectius que ens hem marcat en l'inici d'aquest. En el nostre cas, podem dir que aquest objectius s'han acomplert amb escreix:

- L'objectiu principal de desenvolupar una interfície gràfica per el simulador de nanoestructures BITLLES per tal de realitzar l'entrada de dades que necessitava podem dir que està acomplert.

- A més, tal i com demostra el test que hem realitzat en el capítol anterior de realitzar un disseny real, la nostra interfície gràfica permet realitzar els dissenys amb molta més eficàcia, facilitat i un temps molt menor.

- S'ha aconseguit no duplicar codi font amb el nucli de simulació integrant les comprovacions que realitza aquest a través de la crida a funcions externes que es realitza des de la nostra interfície.

- També hem aconseguit que l'aplicació fos multiplataforma i funciones tan en plataformes GNU/Linux com en les diverses versions del sistema Microsoft Windows, tal i com es demostra en el test multiplataforma explicat al capítol anterior.

- Finalment, i com una de les funcionalitats més destacables del projecte, el fet que els desenvolupadors del nucli de simulació puguin seguir ampliant les seves funcionalitats sense preocupar-se de l'adaptació de la interfície a aquestes, és una gran avantatge que els facilitarà molt la seva feina en un futur.

Per tot això podem concloure doncs, que el projecte ha estat un èxit ja que hem assolit tots els objectius marcats inicialment.

Un altre element important a analitzar és quin nivell de compliment s'ha tingut de la planificació inicial. Com podem veure a la figura 6.1 en termes generals la planificació s'ha complert força. Els primers 3 mesos del projecte, la planificació es pot dir que es va complir perfectament sobretot pel fet que la càrrega de feina era inferior a la dels següents 3 mesos i que les tasques s'havien programat amb un marge força elevat de temps. En quan al segon tram de la planificació han sorgit alguns retrassos sobre el què es tenia pensat. En primer lloc, cal destacar el fet que la codificació de la part 3D es va allargar 1 setmana més de l'esperat. Una altra tasca que es va allargar fou la de la codificació de la comunicació nucli-interfície a causa de la seva dificultat i alguns problemes inesperats per la poca coneixença del llenguatge FORTRAN. Tot això va portar a un retardament en el començament de la redacció de la memòria. No fou un problema greu ja que la idea inicial era començar la memòria aviat per poder fer-la amb calma. Finalment s'ha realitzat amb menys marge de temps però la tasca ha pogut finalitzar abans del termini màxim que tenia. Un altre efecte de tot això és el fet retardar la depuració dels errors del programa i fer-la en part, en paral·lel a la preparació de la presentació. Tot i així, cal valorar positivament la planificació feta ja que es va fer amb els marges suficients com per poder absorbir els retrassos que finalment han sorgit.

Figura 6.1: Comparativa entre la planificació inicial i la real

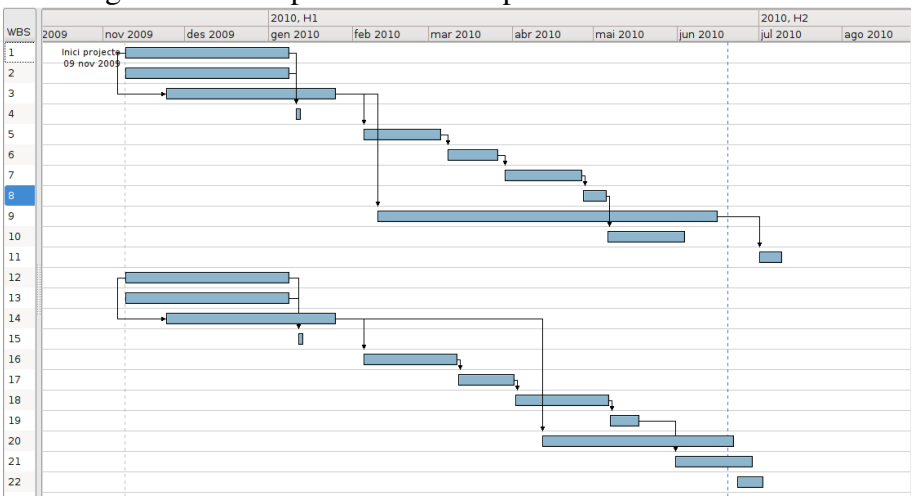


Figura 6.2: Llistat de les tasques de la planificació

WBS	Nom
1	Anàlisi funcional i de requeriments
2	Investigació de l'estat de l'art
3	Disseny de l'aplicació
4	Redacció de l'informe previ
5	Codificació de la interfície 3D
6	Codificació de la interfície de configuració dels parametres de simulació
7	Codificació de la comunicació nucli - interfície
8	Codificació de desar/carregar dissenys
9	Redacció de la memòria
10	Depuració i correcció d'errors en l'aplicació
11	Presentació del projecte
12	Anàlisi funcional i de requeriments REAL
13	Investigació de l'estat de l'art REAL
14	Disseny de l'aplicació REAL
15	Redacció de l'informe previ REAL
16	Codificació de la interfície 3D REAL
17	Codificació de la interfície de configuració dels parametres de simulació REAL
18	Codificació de la comunicació nucli - interfície REAL
19	Codificació de desar/carregar dissenys REAL
20	Redacció de la memòria REAL
21	Depuració i correcció d'errors en l'aplicació REAL
22	Presentació del projecte REAL

6.2 Futur i proposta d'ampliacions

Com s'ha dit a l'apartat anterior hem complert tots els objectius marcats inicialment. Actualment tenim una interfície gràfica que ens permet la realització de dissenys de nanoestructures electròniques pel nucli de simulació, que a més, és capaç d'adaptar-se a possibles ampliacions en el nucli de simulació, cosa que li confereix un gran avantatge i la fa apte pel horitzó a mitjà termini que té el projecte BITLLES.

De fet, tot just ara aquest projecte està començant a promocionar-se en diverses conferències com per exemple a la *14th International Workshop on Computational Electronics* (IWCE 2010) [29] que es fa aquesta tardor a Pisa.

Podem conclouré doncs, que el simulador, en un futur, tindrà un ús molt intens cosa que farà que possiblement si trobi alguna mancança en el seu funcionament, o bé la necessitat d'alguna nova funcionalitat. De fet, durant l'etapa de l'estudi de la viabilitat, es van descartar algunes propostes de funcionalitat pel fet de superar el temps de desenvolupament del que es disposava. Una d'elles era la possibilitat d'exportar imatges dels dissenys realitzat amb l'aplicació a format Postcript. Aquesta funcionalitat seria d'utilitat pel fet que els usuaris de l'aplicació poden estar interessats en la confecció d'articles per a revistes científiques. Aquestes revistes acostumen a treballar amb imatges amb aquest format i això permetria als usuaris de l'aplicació afegir als seus articles figures que els il·lustressin.

A causa de tot això, caldrà que segueixi tenint una vinculació amb el projecte per tal de col·laborar en el seu perfeccionament i fer possible el màxim èxit i expansió d'aquest.

Bibliografia

[1] X. Oriols, “Quantum trajectory approach to time dependent transport in mesoscopic systems with electron-electron interactions”, *Phys. Rev. Let.* 98(6), 066803-066807, (2007).

[2] Intl. Technology Roadmap for Semiconductors, 2007 Edition (and 2008 update).
<http://www.itrs.net/home.html>

[3] ENIAC Strategic Research Agenda 2007.
<http://www.eniac.eu/>

[4] Damocles, IBM.
<http://www.research.ibm.com/DAMOCLES/>

[5] Silvaco.
<http://www.silvaco.com/>

[6] Synopsys.
<http://www.synopsys.com/>

[7] Nemo - 3D, Nanoelectronic Modeling Group.
<https://engineering.purdue.edu/gekcogrp/software-projects/nemo3D/>

[8] nanoHUB.
<http://www.nanohub.org/>

[9] nextnano.
<http://www.nextnano.de/>

[10] Tiber Cad.
<http://www.tibercad.org/>

[11] Jeremy Reimer. "A History of the GUI". Ars Technica. May 5, 2005.
<http://arstechnica.com/old/content/2005/05/gui.ars/3/>

[12] Cocoa, Apple Inc.
<http://developer.apple.com/technologies/mac/cocoa.html>

[13] GTK+.
<http://www.gtk.org/>

[14] One Laptop Per Child Project.
<http://www.laptop.org/en/>

[15] DirectX, Wikipedia.
<http://en.wikipedia.org/wiki/DirectX/>

[16] OpenGL, Wikipedia.
<http://en.wikipedia.org/wiki/OpenGL/>

[17] Swing (Java), Wikipedia.
<http://en.wikipedia.org/wiki/Swing_java/>

[18] Qt (framework), Wikipedia.
<http://en.wikipedia.org/wiki/Qt_(framework)/>

[19] Microsoft Foundation Class Library, Wikipedia.
<http://en.wikipedia.org/wiki/Microsoft_Foundation_Classes/>

[20] MeeGo.
<http://meego.com/>

[21] Comparison of OpenGL and Direct3D, Wikipedia.
<http://en.wikipedia.org/wiki/Comparison_of_OpenGL_and_Direct3D/>

[22] Gtk+ OpenGL Extension, GNOME.
<http://projects.gnome.org/gtkglext/>

[23] WebGL - OpenGL ES 2.0 for the Web, Khronos Group.
<http://www.khronos.org/webgl/>

[24] Web 2.0, Wikipedia.
<http://en.wikipedia.org/wiki/Web_2.0/>

[25] GNU Lesser General Public License, GNU.
<http://www.gnu.org/copyleft/lesser.html>

[26] GNU General Public License, GNU.
<http://www.gnu.org/licenses/gpl.html>

[27] Qt Licensing, Nokia.
<http://qt.nokia.com/products/licensing/licensing/>

[28] Tutorial: Using C/C++ and Fortran together, Yolinux.com.
<http://www.yolinux.com/TUTORIALS/LinuxTutorialMixingFortranAndC.html>

[29] 14th International Workshop on Computational Electronics.
<http://paine.iet.unipi.it/iwce2010/?q=node/1>

Firmat:

Bellaterra, juny de 2010

Resum

L'emergent indústria nanoelectrònica necessita de nous simuladors de dispositius nanoelectrònics. Des del departament d'Enginyeria Electrònica de la UAB se n'està desenvolupant un anomenat BITLLES. Malgrat el seu avançat estat de desenvolupament els manca d'una interfície gràfica que faciliti l'entrada de dades al simulador. Aquest projecte final de carrera s'encarregarà de confeccionar aquesta peça clau per tal de assolir una aplicació prou madura per la seva comercialització. A partir de les tecnologies existents per a la realització d'interfícies es farà una selecció de les eines més adequades pels nostres objectius i seguidament es dissenyarà la interfície gràfica adaptant-la a les necessitats i objectius dels desenvolupadors del projecte BITLLES i dels possibles futurs usuaris.

Resumen

La emergente industria nanoelectrónica necesita de nuevos simuladores de dispositivos nanoelectrónicos. Desde el departamento de Enginyeria Electrónica de la UAB se está desarrollando uno cuyo nombre es BITLLES. Aunque su estado de desarrollo está muy avanzado les falta una interficie gráfica que facilite la entrada de datos al simulador. Este proyecto de final de carrera se encargará de realizar esta pieza clave para conseguir una aplicación lo suficiente madura para su comercialización. A partir de las tecnologías existentes para la realización de interficies se hará una selección de las herramientas mas adecuadas para nuetros objetivos y seguidamente se diseñará la interficie gráfica adaptandola a las necesidades y objetivos de los desarrolladores del proyecto BITLLES y a sus posibles futuros usuarios.

Abstract

The emerging nanoelectronics industry needs new nanoelectronic device simulator. The department of Enginyeria Electrònica at UAB is developing one called BITLLES. Despite its advanced state of development it lacks a graphical interface that facilitates the data input to the simulator. This final degree project will prepare this key piece to achieve an application mature enough to be marketed. On the basis of existing technologies for the realization of interfaces we will do a selection of the tools suitable for our purposes and then we will design the graphical interface adapted to the needs and objectives of the project developers and the potential future users of BITLLES.